VSC7430 Datasheet
6-Port Carrier Ethernet Switch with ViSAA™, VeriTime™, and Gigabit Ethernet PHYs



Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

©2019 Microsemi, a wholly owned subsidiary of Microchip Technology Inc. All rights reserved. Microsemi and the Microsemi logo are registered trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.

Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.

Contents

1 Revision History . . . 1
1.1 Revision 4.1 . . . 1
1.2 Revision 4.0 . . . 1
2 Product Overview . . . 2
2.1 General Features . . . 2
2.1.1 Layer 2 and Layer 3 Forwarding . . . 2
2.1.2 Carrier Ethernet Support . . . 2
2.1.3 Timing and Synchronization . . . 3
2.1.4 Quality of Service (QoS) . . . 3
2.1.5 Security . . . 3
2.1.6 Management . . . 3
2.1.7 Product Parameters . . . 4
2.2 Applications . . . 5
2.2.1 Wireless Backhaul . . . 5
2.2.2 Network Interface Device (NID) . . . 8
2.2.3 Carrier Ethernet Switch with MPLS-TP . . . 11
2.2.4 Small Cell Application . . . 11
3 Functional Descriptions . . . 13
3.1 Register Notations . . . 13
3.2 Functional Overview . . . 14
3.2.1 Frame Arrival in Ports and Port Modules . . . 15
3.2.2 Basic Classification . . . 15
3.2.3 Virtualized Service Aware Architecture (ViSAA™) . . . 15
3.2.4 Security and Control Protocol Classification . . . 16
3.2.5 Policing . . . 16
3.2.6 Layer 2 Forwarding . . . 17
3.2.7 Layer 3 Forwarding . . . 18
3.2.8 Shared Queue System and Hierarchical Scheduler . . . 18
3.2.9 Rewriter and Frame Departure . . . 20
3.2.10 CPU Port Module . . . 21
3.2.11 Synchronous Ethernet and Precision Time Protocol (PTP) . . . 21
3.2.12 Ethernet and MPLS OAM . . . 22
3.2.13 CPU Subsystem . . . 22
3.3 Frame Headers . . . 22
3.3.1 Internal Frame Header Placement . . . 23
3.3.2 Internal Frame Header Layout . . . 23
3.3.3 VStaX Header . . . 29
3.4 Port Numbering and Mappings . . . 33
3.4.1 Supported SerDes Interfaces . . . 34
3.4.2 Dual-Media Mode . . . 34
3.4.3 Logical Port Numbers . . . 34
3.5 SERDES1G . . . 34
3.6 SERDES6G . . . 35
3.7 Copper Transceivers . . . 35
3.7.1 Register Access . . . 35
3.7.2 Cat5 Twisted Pair Media Interface . . . 36
3.7.3 Wake-On-LAN and SecureOn . . . 39
3.7.4 Ethernet Inline Powered Devices . . . 39
3.7.5 IEEE 802.3af PoE Support . . . 40

3.7.6 ActiPHY™ Power Management . . . 41
3.7.7 Testing Features . . . 42
3.7.8 VeriPHY™ Cable Diagnostics . . . 43
3.8 DEV1G and DEV2G5 Port Modules . . . 44
3.8.1 MAC . . . 44
3.8.2 Half-Duplex Mode . . . 45
3.8.3 Physical Coding Sublayer (PCS) . . . 46
3.8.4 Port Statistics . . . 49
3.9 Assembler . . . 49
3.9.1 Setting Up a Port in the Assembler . . . 50
3.9.2 Setting Up a Port for Frame Injection . . . 50
3.9.3 Setting Up MAC Control Sublayer PAUSE Frame Detection . . . 51
3.9.4 Setting Up PFC . . . 52
3.9.5 Setting Up Assembler Port Statistics . . . 52
3.9.6 Setting Up the Loopback Path . . . 53
3.10 Versatile Content-Aware Processor (VCAP) . . . 54
3.10.1 Configuring VCAP . . . 54
3.10.2 Wide VCAP Entries and Actions . . . 57
3.10.3 Individual VCAPs . . . 58
3.10.4 VCAP Programming Examples . . . 64
3.11 Pipeline Points . . . 66
3.11.1 Pipeline Definitions . . . 67
3.12 Analyzer . . . 69
3.12.1 Initializing the Analyzer . . . 70
3.13 VCAP CLM Keys and Actions . . . 70
3.13.1 Keys Overview . . . 70
3.13.2 VCAP CLM X1 Key Details . . . 73
3.13.3 VCAP CLM X2 Key Details . . . 74
3.13.4 VCAP CLM X4 Key Details . . . 77
3.13.5 VCAP CLM X8 Key Details . . . 80
3.13.6 VCAP CLM X16 Key Details . . . 86
3.13.7 VCAP CLM Actions . . . 91
3.14 Analyzer Classifier . . . 102
3.14.1 Basic Classifier . . . 102
3.14.2 VCAP CLM Processing . . . 115
3.14.3 QoS Mapping Table . . . 126
3.14.4 Ingress Protection Table (IPT) . . . 127
3.14.5 Layer 2 Control Protocol Processing . . . 128
3.14.6 Y.1731 Ethernet MIP . . . 129
3.14.7 Analyzer Classifier Diagnostics . . . 130
3.15 VLAN and MSTP . . . 130
3.15.1 Private VLAN . . . 131
3.15.2 VLAN Pseudo Code . . . 132
3.16 VCAP LPM: Keys and Action . . . 137
3.16.1 VCAP LPM SGL_IP4 Key Details . . . 138
3.16.2 VCAP LPM DBL_IP4 Key Details . . . 138
3.16.3 VCAP LPM SGL_IP6 Key Details . . . 138
3.16.4 VCAP LPM DBL_IP6 Key Details . . . 138
3.16.5 VCAP LPM Actions . . . 138
3.17 IP Processing . . . 140
3.17.1 IP Source/Destination Guard . . . 140
3.17.2 IP Routing . . . 142
3.17.3 Statistics . . . 154
3.17.4 IGMP/MLD Snooping Switch . . . 156
3.18 VCAP IS2 Keys and Actions . . . 156
3.18.1 VCAP IS2 Keys . . . 157
3.18.2 VCAP IS2 Actions . . . 163

3.19 Analyzer Access Control Lists . . . 166
3.19.1 VCAP IS2 . . . 167
3.19.2 Analyzer Access Control List Frame Rewriting . . . 173
3.20 Analyzer Layer 2 Forwarding and Learning . . . 176
3.20.1 Analyzer MAC Table . . . 176
3.20.2 MAC Table Updates . . . 178
3.20.3 CPU Access to MAC Table . . . 178
3.20.4 SCAN Command . . . 182
3.20.5 Forwarding Lookups . . . 184
3.20.6 Source Check and Automated Learning . . . 186
3.20.7 Automated Aging (AUTOAGE) . . . 189
3.20.8 Service Handling . . . 191
3.20.9 Interrupt Handling . . . 193
3.21 Analyzer Access Control Forwarding, Policing, and Statistics . . . 193
3.21.1 Mask Handling . . . 193
3.21.2 Policing . . . 198
3.21.3 Analyzer Statistics . . . 207
3.21.4 Analyzer sFlow Sampling . . . 217
3.21.5 Mirroring . . . 218
3.22 Versatile OAM Processor (VOP) . . . 220
3.22.1 VOP Blocks . . . 220
3.22.2 Versatile OAM Endpoint (VOE) Functions . . . 221
3.22.3 Supported OAM PDUs . . . 222
3.22.4 VOE Locations . . . 222
3.23 VOP Common Functions . . . 225
3.23.1 Accessing the VOP . . . 225
3.23.2 VOE Hierarchy . . . 225
3.23.3 VOE Frame Injection and Extraction . . . 228
3.23.4 Loss Measurement (LM) Counters . . . 230
3.23.5 Port Count-All Rx/Tx Counters . . . 232
3.23.6 Basic VOP Configuration . . . 233
3.23.7 Loss of Continuity Controller . . . 233
3.23.8 Hit-Me-Once Controller . . . 234
3.23.9 Interrupt Controller . . . 236
3.23.10 Ethernet Configuration . . . 237
3.23.11 MPLS-TP Configuration . . . 238
3.24 VOE: Ethernet OAM . . . 239
3.24.1 Ethernet VOE Functions . . . 239
3.24.2 Continuity Check Messages (CCM) . . . 246
3.24.3 VOE LOCC Configuration . . . 248
3.24.4 Test Frames (TST) . . . 249
3.24.5 Loopback Frames (LBM/LBR) . . . 250
3.24.6 Frame Loss Measurement, Single-Ended . . . 251
3.24.7 Frame Loss Measurement, Dual-Ended . . . 253
3.24.8 Synthetic Loss Measurement . . . 254
3.24.9 Synthetic Loss Measurement, Single-Ended . . . 260
3.24.10 Synthetic Loss Measurement, Dual-Ended . . . 261
3.24.11 Delay Measurement . . . 262
3.24.12 Single-Ended Delay Measurement (SE-DM: DMM/DMR) . . . 262
3.24.13 Dual-Ended Delay Measurement (DE-DM: 1DM) . . . 263
3.24.14 Generic/Unknown Opcodes . . . 264
3.24.15 Link Trace . . . 265
3.24.16 Non-OAM Sequence Numbering . . . 265
3.24.17 Service Activation Test (SAT) . . . 270
3.24.18 G.8113.1 Specific Functions . . . 272
3.25 VOE: MPLS-TP OAM . . . 274
3.25.1 MPLS-TP VOE Functions . . . 274
3.25.2 Bidirectional Forwarding Detection (BFD) Implementation . . . 275

3.25.3 BFD Functional Overview . . . 275
3.25.4 BFD Configuration . . . 277
3.25.5 BFD Frame Reception . . . 278
3.25.6 BFD Frame Transmission . . . 282
3.25.7 BFD VOE Functions . . . 283
3.25.8 BFD Statistics . . . 284
3.26 Shared Queue System and Hierarchical Scheduler . . . 284
3.26.1 Analyzer Result . . . 286
3.26.2 Buffer Control . . . 287
3.26.3 Forwarding . . . 287
3.26.4 Congestion Control . . . 290
3.26.5 Queue Mapping . . . 296
3.26.6 Queue Congestion Control . . . 299
3.26.7 Scheduling . . . 301
3.26.8 Queue System Initialization . . . 305
3.26.9 Miscellaneous Features . . . 305
3.27 Automatic Frame Injector . . . 308
3.27.1 Injection Tables . . . 309
3.27.2 Frame Table . . . 310
3.27.3 Delay Triggered Injection . . . 311
3.27.4 Timer Triggered Injection . . . 315
3.27.5 Injection Queues . . . 320
3.27.6 Adding Injection Frame . . . 320
3.27.7 Starting Injection . . . 321
3.27.8 Stopping Injection . . . 321
3.27.9 Removing Injection Frames . . . 321
3.27.10 Port Parameters . . . 321
3.28 Rewriter . . . 322
3.28.1 Rewriter Operation . . . 322
3.28.2 Supported Ports . . . 323
3.28.3 Supported Frame Formats . . . 323
3.28.4 Rewriter Initialization . . . 323
3.28.5 VCAP_ES0 Lookup . . . 323
3.28.6 Mapping Tables . . . 336
3.28.7 VLAN Editing . . . 338
3.28.8 DSCP Remarking . . . 343
3.28.9 VStaX Header Insertion . . . 343
3.28.10 Forwarding to GCPU . . . 344
3.28.11 Layer 3 Routing . . . 346
3.28.12 OAM Frame Handling . . . 346
3.28.13 Mirror Frames . . . 351
3.28.14 MPLS Editing . . . 352
3.28.15 Internal Frame Header Insertion . . . 358
3.28.16 Frame Injection from Internal CPU . . . 359
3.29 Disassembler . . . 359
3.29.1 Setting Up Ports . . . 359
3.29.2 Maintaining the Cell Buffer . . . 360
3.29.3 Setting Up MAC Control Sublayer PAUSE Function . . . 360
3.29.4 Setting up Flow Control in Half-Duplex Mode . . . 361
3.29.5 Setting Up Frame Aging . . . 362
3.29.6 Setting Up Transmit Data Rate Limiting . . . 362
3.29.7 Error Detection . . . 364
3.30 Layer 1 Timing . . . 364
3.31 Hardware Time Stamping . . . 367
3.31.1 One-Step Functions . . . 369
3.31.2 Calculation Overview . . . 370
3.31.3 Detecting Calculation Issues . . . 371
3.31.4 Two-Step Functions . . . 371

3.31.5 Time of Day Time Stamping . . . 372
3.31.6 Time of Day Generation . . . 372
3.31.7 Multiple PTP Time Domains . . . 373
3.31.8 Register Interface to 1588 Functions . . . 373
3.31.9 Configuring I/O Delays . . . 375
3.32 VRAP Engine . . . 375
3.32.1 VRAP Request Frame Format . . . 376
3.32.2 VRAP Response Frame Format . . . 377
3.32.3 VRAP Header Format . . . 377
3.32.4 VRAP READ Command . . . 377
3.32.5 VRAP READ-MODIFY-WRITE Command . . . 378
3.32.6 VRAP IDLE Command . . . 378
3.32.7 VRAP PAUSE Command . . . 378
3.33 Energy Efficient Ethernet . . . 379
3.34 CPU Injection and Extraction . . . 380
3.34.1 Frame Injection . . . 380
3.34.2 Frame Extraction . . . 381
3.34.3 Forwarding to CPU . . . 381
3.34.4 Automatic Frame Injection (AFI) . . . 381
3.35 Priority-Based Flow Control (PFC) . . . 382
3.35.1 PFC Pause Frame Generation . . . 382
3.35.2 PFC Frame Reception . . . 383
3.36 Protection Switching . . . 384
3.36.1 Ethernet Ring Protection Switching . . . 384
3.36.2 Linear Protection Switching for E-Line Services . . . 387
3.36.3 Link Aggregation . . . 390
3.36.4 Port Protection Switching . . . 390
3.37 Clocking and Reset . . . 390
3.37.1 Pin Strapping . . . 390

4 VCore-III System and CPU Interfaces . . . 392
4.1 VCore-III Configurations . . . 393
4.2 Clocking and Reset . . . 393
4.2.1 Watchdog Timer . . . 394
4.3 Shared Bus . . . 394
4.3.1 VCore-III Shared Bus Arbitration . . . 396
4.3.2 Chip Register Region . . . 396
4.3.3 SI Flash Region . . . 397
4.3.4 DDR3/DDR3L Region . . . 398
4.3.5 PCIe Region . . . 398
4.4 VCore-III CPU . . . 398
4.4.1 Little Endian and Big Endian Support . . . 398
4.4.2 Software Debug and Development . . . 398
4.5 External CPU Support . . . 398
4.5.1 Register Access and Multimaster Systems . . . 399
4.5.2 Serial Interface in Slave Mode . . . 399
4.5.3 MIIM Interface in Slave Mode . . . 401
4.5.4 Access to the VCore Shared Bus . . . 403
4.5.5 Mailbox and Semaphores . . . 404
4.6 PCIe Endpoint Controller . . . 405
4.6.1 Accessing Endpoint Registers . . . 406
4.6.2 Enabling the Endpoint . . . 406
4.6.3 Base Address Registers Inbound Requests . . . 408
4.6.4 Outbound Interrupts . . . 408
4.6.5 Outbound Access . . . 409
4.6.6 Power Management . . . 410

4.6.7 Device Reset Using PCIe . . . 411
4.7 Frame DMA . . . 412
4.7.1 DMA Control Block Structures . . . 413
4.7.2 Enabling and Disabling FDMA Channels . . . 414
4.7.3 Channel Counters . . . 415
4.7.4 FDMA Events and Interrupts . . . 416
4.7.5 FDMA Extraction . . . 417
4.7.6 FDMA Injection . . . 417
4.7.7 Manual Mode . . . 418
4.8 VCore-III System Peripherals . . . 420
4.8.1 SI Boot Controller . . . 420
4.8.2 SI Master Controller . . . 422
4.8.3 DDR3/DDR3L Memory Controller . . . 427
4.8.4 Timers . . . 432
4.8.5 UARTs . . . 433
4.8.6 Two-Wire Serial Interface . . . 434
4.8.7 MII Management Controller . . . 437
4.8.8 GPIO Controller . . . 439
4.8.9 Serial GPIO Controller . . . 442
4.8.10 Fan Controller . . . 447
4.8.11 Temperature Sensor . . . 448
4.8.12 Memory Integrity Monitor . . . 448
4.8.13 Interrupt Controller . . . 451

5 Registers . . . 457

6 Electrical Specifications . . . 458
6.1 DC Characteristics . . . 458
6.1.1 Reference Clock . . . 458
6.1.2 PLL Clock Output . . . 458
6.1.3 DDR3/DDR3L SDRAM Interface . . . 458
6.1.4 SERDES1G . . . 459
6.1.5 SERDES6G . . . 460
6.1.6 GPIO, SI, JTAG, and Miscellaneous Signals . . . 461
6.1.7 Thermal Diode . . . 462
6.2 AC Characteristics . . . 463
6.2.1 Reference Clock . . . 463
6.2.2 PLL Clock Outputs . . . 465
6.2.3 SERDES1G . . . 465
6.2.4 SERDES6G . . . 466
6.2.5 Reset Timing . . . 469
6.2.6 MII Management . . . 469
6.2.7 Serial Interface (SI) Boot Master Mode . . . 470
6.2.8 Serial Interface (SI) Master Mode . . . 470
6.2.9 Serial Interface (SI) for Slave Mode . . . 471
6.2.10 DDR SDRAM Interface . . . 473
6.2.11 JTAG Interface . . . 475
6.2.12 Serial Inputs/Outputs . . . 477
6.2.13 Recovered Clock Outputs . . . 477
6.2.14 Two-Wire Serial Interface . . . 478
6.2.15 IEEE 1588 Time Tick Outputs . . . 480
6.3 Current and Power Consumption . . . 480
6.3.1 Current Consumption . . . 480
6.3.2 Power Consumption . . . 481
6.3.3 Power Supply Sequencing . . . 481
6.4 Operating Conditions . . . 481
6.5 Stress Ratings . . . 482

7 Pin Descriptions . . . 483
7.1 Pin Diagram . . . 483
7.2 Pins by Function . . . 484
7.2.1 DDR SDRAM Interface . . . 485
7.2.2 General-Purpose Inputs and Outputs . . . 486
7.2.3 JTAG Interface . . . 486
7.2.4 MII Management Interface . . . 486
7.2.5 Miscellaneous . . . 487
7.2.6 PCI Express Interface . . . 487
7.2.7 Power Supplies and Ground . . . 488
7.2.8 SERDES1G . . . 488
7.2.9 SERDES6G . . . 488
7.2.10 Serial CPU Interface . . . 489
7.2.11 System Clock Interface . . . 489
7.2.12 Twisted Pair Interface . . . 490
7.3 234Pins by Number . . . 492
7.4 Pins by Name . . . 495
8 Package Information . . . 498
8.1 Package Drawing . . . 498
8.2 Thermal Specifications . . . 499
8.3 Moisture Sensitivity . . . 500
9 Design Guidelines . . . 501
9.1 Power Supplies . . . 501
9.2 Power Supply Decoupling . . . 501
9.2.1 Reference Clock . . . 501
9.2.2 Single-Ended REFCLK Input . . . 501
9.3 Interfaces . . . 502
9.3.1 General Recommendations . . . 502
9.3.2 SerDes Interfaces (SGMII, 2.5G) . . . 503
9.3.3 Serial Interface . . . 503
9.3.4 PCI Express Interface . . . 503
9.3.5 Two-Wire Serial Interface . . . 504
9.3.6 DDR3 SDRAM Interface . . . 504
9.3.7 Thermal Diode External Connection . . . 505
10 Design Considerations . . . 507
11 Ordering Information . . . 508

Figures

Figure 1 Converged Access and Aggregation Network Application . . . 5
Figure 2 Wireless Backhaul Application . . . 7
Figure 3 E-Access NID Application . . . 8
Figure 4 S-Tagged NID Application for Multi-Operator Access . . . 9
Figure 5 Virtual NID Application for Multi-Operator Access . . . 9
Figure 6 NID Internal View . . . 10
Figure 7 Carrier Ethernet Switch with MPLS-TP Application . . . 11
Figure 8 4G/LTE Small Cell Application . . . 12
Figure 9 RTL Block Diagram . . . 13
Figure 10 Block Diagram . . . 14
Figure 11 Default Scheduler-Shaper Configuration . . . 19
Figure 12 Hierarchical Scheduler-Shaper Configuration . . . 20
Figure 13 Frame with Internal Frame Header . . . 23
Figure 14 Internal Frame Header . . . 24
Figure 15 Frame With VStaX Header . . . 29
Figure 16 VStaX Header Layout . . . 30
Figure 17 Register Space Layout . . . 36
Figure 18 Cat5 Media Interface . . . 37
Figure 19 Low Power Idle Operation . . . 38
Figure 20 Wake-On-LAN Functionality . . . 39
Figure 21 Inline Powered Ethernet Switch . . . 40
Figure 22 ActiPHY State Diagram . . . 41
Figure 23 Far-End Loopback Diagram . . . 43
Figure 24 Near-End Loopback Diagram . . . 43
Figure 25 Connector Loopback Diagram . . . 43
Figure 26 Frame Injection Formats . . . 51
Figure 27 VCAP Cache Layout Example . . . 55
Figure 28 VCAP Cache Type-Group Example . . . 58
Figure 29 Processing Flow . . . 67
Figure 30 VLAN Acceptance Filter . . . 105
Figure 31 Basic QoS Classification Flow Chart . . . 107
Figure 32 Basic DP Classification Flow Chart . . . 108
Figure 33 Basic DSCP Classification Flow Chart . . . 109
Figure 34 Basic VLAN Classification Flow Chart . . . 111
Figure 35 MPLS OAM Detection . . . 122
Figure 36 Example of QoS Mappings . . . 127
Figure 37 VLAN Table Update Engine (TUPE) . . . 136
Figure 38 Router Model . . . 142
Figure 39 Unicast Routing Table Overview . . . 143
Figure 40 Multicast Routing Table Overview . . . 144
Figure 41 Ingress Router Leg Lookup Flow . . . 145
Figure 42 Ingress Router Leg MAC Address Matching for Unicast Packets . . . 145
Figure 43 IP Unicast Routing Example . . . 148
Figure 44 ARP Pointer Remapping . . . 150
Figure 45 IP Multicast Routing Example . . . 152
Figure 46 MAC Table Organization . . . 177
Figure 47 DMAC Lookup . . . 185
Figure 48 Source Check . . . 187
Figure 49 PGID Layout . . . 194
Figure 50 PGID Lookup Decision Forwarding . . . 195
Figure 51 GLAG Port of Exit Calculation . . . 197
Figure 52 Port Mask Operation . . . 198
Figure 53 Policer Hierarchy . . . 200
Figure 54 Sticky Events Available as Global Events . . . 208

Figure 55 Port Statistics Counters . . . 208
Figure 56 Service Statistics Counters . . . 211
Figure 57 Queue Statistics . . . 212
Figure 58 Bundle Policer Statistics . . . 213
Figure 59 BUM Policer Statistics . . . 214
Figure 60 ACL Policer Statistics . . . 215
Figure 61 Ingress and Egress Routing Statistics per Router Leg per IP Version . . . 216
Figure 62 sFlow Stamp Format in FCS . . . 218
Figure 63 Ingress Mirroring in Specific VLAN . . . 220
Figure 64 Down-MEP Location . . . 223
Figure 65 Down-VOE Location . . . 224
Figure 66 Up-MEP Location . . . 224
Figure 67 Up-VOE Location . . . 225
Figure 68 Down-MEP VOE Hierarchy . . . 226
Figure 69 Up-MEP VOE hierarchy . . . 227
Figure 70 PDU Modification Location . . . 230
Figure 71 Ethernet VOE: Tx Validation . . . 240
Figure 72 Rx Validation, Part 1 . . . 241
Figure 73 Rx Validation, Part 2 . . . 244
Figure 74 Single-Ended SynLM . . . 255
Figure 75 Dual-Ended SynLM . . . 256
Figure 76 SynLM Tx Validation . . . 257
Figure 77 SLM/SL1 Rx Validation . . . 258
Figure 78 SLR Rx Validation . . . 259
Figure 79 SLR Updates . . . 261
Figure 80 Rx 1SL Updates . . . 261
Figure 81 Single-Ended SAM_SEQ Frame Flow . . . 266
Figure 82 Dual-Ended SAM_SEQ Frame Flow . . . 267
Figure 83 SAM_SEQ Number Format . . . 268
Figure 84 Standard VOE Hierarchy . . . 271
Figure 85 SAT VOE Hierarchy . . . 272
Figure 86 BFD Coordinated Mode . . . 275
Figure 87 BFD Independent Mode . . . 276
Figure 88 Shared Queue System Block Diagram . . . 285
Figure 89 Translation of Transmit Requests . . . 289
Figure 90 Accounting Sheet Example . . . 291
Figure 91 Reserved and Shared Resource Overview . . . 291
Figure 92 Strict Priority Sharing . . . 292
Figure 93 Per Priority Sharing . . . 292
Figure 94 WRED Sharing . . . 293
Figure 95 WRED Profiles . . . 294
Figure 96 Scheduler Hierarchy (Normal Scheduling Mode) . . . 296
Figure 97 Queue Mapping Tables . . . 297
Figure 98 Group Scheduling Mode . . . 298
Figure 99 Mobile Backhaul Mode . . . 298
Figure 100 Drop Decision Flow . . . 299
Figure 101 Queue Limitation Share . . . 300
Figure 102 Default Scheduling Hierarchy . . . 302
Figure 103 Scheduler . . . 303
Figure 104 Internal Bus . . . 307
Figure 105 Injection Tables . . . 310
Figure 106 DTI Frame/Delay Sequence Example . . . 312
Figure 107 Fine Tuning Bandwidth of DTI Sequence . . . 312
Figure 108 Fine Tuning Bandwidth of Multiframe DTI Sequence . . . 313
Figure 109 DTI Frame/Delay Sequence Using Inject Forever . . . 313
Figure 110 DTI Concatenation . . . 314
Figure 111 TTI Calendar Example . . . 317
Figure 112 AFI TUPE . . . 319
Figure 113 Frame Forward Options . . . 333

Figure 114 Mapping Tables . . . 337
Figure 115 VLAN Pushing . . . 339
Figure 116 VLAN Tag Construction . . . 341
Figure 117 MPLS Encapsulation Data . . . 353
Figure 118 MPLS Remarking . . . 357
Figure 119 Supported KEEP_IFH_SEL Formats . . . 358
Figure 120 Supported Time Stamping Flows . . . 367
Figure 121 Frame Flow . . . 368
Figure 122 Time Stamp Bus and FIFO . . . 371
Figure 123 Timing Distribution . . . 372
Figure 124 VRAP Request Frame Format . . . 376
Figure 125 VRAP Header Format . . . 377
Figure 126 READ Command . . . 377
Figure 127 WRITE Command . . . 378
Figure 128 READ-MODIFY-WRITE Command . . . 378
Figure 129 IDLE Command . . . 378
Figure 130 PAUSE Command . . . 378
Figure 131 EEE State Diagram . . . 379
Figure 132 Up-MEP Injection from AFI . . . 382
Figure 133 Down-MEP Injection from AFI . . . 382
Figure 134 PFC Generation Per Port . . . 383
Figure 135 Ethernet Ring Protection Example . . . 385
Figure 136 Ethernet Ring with Failure . . . 386
Figure 137 VCore-III System Block Diagram . . . 392
Figure 138 Shared Bus Memory Map . . . 395
Figure 139 Chip Registers Memory Map . . . 397
Figure 140 SI Slave Mode Register . . . 399
Figure 141 Write Sequence for SI . . . 400
Figure 142 Read Sequence for SI_CLK Slow . . . 401
Figure 143 Read Sequence for SI_CLK Pause . . . 401
Figure 144 Read Sequence for One-Byte Padding . . . 401
Figure 145 MIIM Slave Write Sequence . . . 403
Figure 146 MIIM Slave Read Sequence . . . 403
Figure 147 FDMA DCB Layout . . . 414
Figure 148 FDMA Channel States . . . 415
Figure 149 FDMA Channel Interrupt Hierarchy . . . 417
Figure 150 Extraction Status Word Encoding . . . 419
Figure 151 Injection Status Word Encoding . . . 420
Figure 152 SI Boot Controller Memory Map in 24-Bit Mode . . . 421
Figure 153 SI Boot Controller Memory Map in 32-Bit Mode . . . 421
Figure 154 SI Read Timing in Normal Mode . . . 422
Figure 155 SI Read Timing in Fast Mode . . . 422
Figure 156 SIMC SPI Clock Configurations . . . 425
Figure 157 SIMC SPI 3x Transfers . . . 425
Figure 158 Memory Controller ODT Hookup . . . 430
Figure 159 UART Timing . . . 433
Figure 160 Two-Wire Serial Interface Timing for 7-bit Address Access . . . 436
Figure 161 MII Management Timing . . . 438
Figure 162 SIO Timing . . . 443
Figure 163 SIO Timing with SGPIOs Disabled . . . 444
Figure 164 SGPIO Output Order . . . 444
Figure 165 Link Activity Timing . . . 446
Figure 166 Monitor State Diagram . . . 450
Figure 167 Memory Detection Logic . . . 451
Figure 168 Interrupt Source Logic . . . 454
Figure 169 Interrupt Destination Logic . . . 455
Figure 170 Port Module Interrupt Logic . . . 455
Figure 171 Thermal Diode . . . 462
Figure 172 REFCLK Jitter Transfer Curves . . . 464

Figure 173 nRESET Signal Timing Specifications . . . 469
Figure 174 MIIM Timing Diagram . . . 469
Figure 175 SI Timing Diagram for Master Mode . . . 471
Figure 176 SI Input Data Timing Diagram for Slave Mode . . . 472
Figure 177 SI Output Data Timing Diagram for Slave Mode . . . 472
Figure 178 SI_DO Disable Test Circuit . . . 473
Figure 179 DDR SDRAM Input Timing Diagram . . . 473
Figure 180 DDR SDRAM Output Timing Diagram . . . 474
Figure 181 Test Load Circuit for DDR3 Outputs . . . 475
Figure 182 JTAG Interface Timing Diagram . . . 476
Figure 183 Test Circuit for TDO Disable Time . . . 476
Figure 184 Serial I/O Timing Diagram . . . 477
Figure 185 Test Circuit for Recovered Clock Output Signals . . . 478
Figure 186 Two-Wire Serial Read Timing Diagram . . . 478
Figure 187 Two-Wire Serial Write Timing Diagram . . . 479
Figure 188 Pin Diagram, Top Left . . . 483
Figure 189 Pin Diagram, Top Right . . . 484
Figure 190 Package Drawing . . . 499
Figure 191 2.5 V CMOS Single-Ended REFCLK Input Resistor Network . . . 502
Figure 192 3.3 V CMOS Single-Ended REFCLK Input Resistor Network . . . 502
Figure 193 16-Bit DDR3 SDRAM Point-to-Point Routing . . . 504
Figure 194 External Temperature Monitor Connection . . . 506

Tables

Table 1 Product Parameters . . . 4
Table 2 Internal Frame Header Fields . . . 24
Table 3 VStaX Header Fields . . . 31
Table 4 Default Port Numbering and Port Mappings . . . 33
Table 5 Interface Macro to I/O Pin Mapping . . . 34
Table 6 Supported SerDes Interfaces . . . 34
Table 7 Supported MDI Pair Combinations . . . 38
Table 8 DEV1G and DEV2G5 MAC Configuration Registers Overview . . . 44
Table 9 DEV1G and DEV2G5 Reset . . . 44
Table 10 DEV1G and DEV2G5 Interrupt Sources Register Overview . . . 45
Table 11 DEV1G and DEV2G5 Port Mode Configuration Registers Overview . . . 45
Table 12 DEV1G and DEV2G5 PCS Configuration Registers Overview . . . 46
Table 13 DEV1G and DEV2G5 PCS Test Pattern Configuration Registers . . . 48
Table 14 DEV1G and DEV2G5 PCS EEE Configuration Registers Overview . . . 48
Table 15 DEV1G and DEV2G5 100BASE-FX Configuration Registers Overview . . . 49
Table 16 Port Configuration Register Overview . . . 50
Table 17 Port Status Register Overview . . . 50
Table 18 Injection VLAN Register Overview . . . 51
Table 19 PAUSE Frame Detection Configuration Register Overview . . . 51
Table 20 PFC Configuration Register Overview . . . 52
Table 21 Port Statistics Configuration Register Overview . . . 52
Table 22 Loopback FIFO Configuration Register Overview . . . 53
Table 23 Loopback FIFO Sticky-Bit Registers Overview . . . 53
Table 24 VCAP_ES0 and VCAP_SUPER Registers . . . 54
Table 25 Entry Bit Encoding . . . 55
Table 26 VCAP CLM Parameters . . . 59
Table 27 VCAP CLM Entry Type-Group Fields . . . 59
Table 28 VCAP CLM Action Type-Group Fields . . . 59
Table 29 VCAP CLM Entry/Action Bit Distribution . . . 60
Table 30 VCAP CLM Default Rule Addresses . . . 60
Table 31 VCAP IS2 Parameters . . . 61
Table 32 VCAP IS2 Entry Type-Group Fields . . . 61
Table 33 VCAP IS2 Entry/Action Bit Distribution . . . 61
Table 34 VCAP LPM Parameters . . . 62
Table 35 VCAP LPM Entry Type-Group Fields . . . 62
Table 36 VCAP LPM Entry/Action Bit Distribution . . . 63
Table 37 VCAP ES0 Parameters . . . 63
Table 38 VCAP ES0 Entry/Action Bit Distribution . . . 64
Table 39 CLM X2 Entry Type-Group and Distribution Summary . . . 64
Table 40 CLM X4 Action Type-Group and Distribution Summary . . . 64
Table 41 Pipeline Definitions . . . 67
Table 42 Pipeline Actions . . . 69
Table 43 VCAP CLM Keys and Sizes . . . 70
Table 44 VCAP CLM Key Overview . . . 71
Table 45 VCAP CLM X1 Key Details . . . 73
Table 46 VCAP CLM X2 Key Details . . . 74
Table 47 VCAP CLM X4 Key Details . . . 77
Table 48 VCAP CLM X8 Key Details . . . 80
Table 49 VCAP CLM X16 Key Details . . . 86
Table 50 VCAP CLM Action Selection . . . 91
Table 51 VCAP CLM Actions . . . 91
Table 52 VLAN Tag Extraction Register . . . 103
Table 53 Routing Tag Control Register . . . 104
Table 54 Frame Acceptance Filtering Registers . . . 104

Table 55 QoS, DP, and DSCP Classification Registers . . . 106
Table 56 VLAN Configuration Registers . . . 109
Table 57 Aggregation Code Generation Registers . . . 112
Table 58 CPU Forwarding Registers . . . 112
Table 59 Frame Type Definitions for CPU Forwarding . . . 113
Table 60 Port VOE Tag Handling Registers Overview . . . 114
Table 61 Port Configuration of VCAP CLM . . . 115
Table 62 Reserved Label Skipping Configuration . . . 117
Table 63 Miscellaneous VCAP CLM Key Configuration . . . 124
Table 64 VCAP CLM Range Checker Configuration Registers Overview . . . 125
Table 65 VCAP CLM QoS Mapping Table Registers Overview . . . 126
Table 66 VLAN Table (4224 Entries) . . . 130
Table 67 MSTP Table (66 Entries) . . . 131
Table 68 Common VLAN and MSTP Parameters . . . 131
Table 69 VLAN Table Update Engine (TUPE) Parameters . . . 135
Table 70 VCAP LPM Key and Sizes . . . 137
Table 71 VCAP LPM Key Overview . . . 137
Table 72 VCAP LPM SGL_IP4 Key Details . . . 138
Table 73 VCAP LPM DBL_IP4 Key Details . . . 138
Table 74 VCAP LPM SGL_IP6 Key Details . . . 138
Table 75 VCAP LPM DBL_IP6 Key Details . . . 138
Table 76 VCAP LPM Action Selection . . . 139
Table 77 VCAP LPM Actions . . . 139
Table 78 CPU Queue Overview . . . 153
Table 79 Ingress Router Leg Events . . . 154
Table 80 Ingress Router Leg Statistics Example . . . 155
Table 81 Egress Router Leg Events . . . 155
Table 82 Egress Router Leg Statistics Example . . . 156
Table 83 VCAP IS2 Keys and Sizes . . . 157
Table 84 VCAP IS2 Key Overview . . . 157
Table 85 VCAP IS2 Actions . . . 163
Table 86 VCAP IS2 Frame Types . . . 167
Table 87 Port Configuration of VCAP IS2 . . . 168
Table 88 VCAP IS2 Key Selection . . . 168
Table 89 VCAP IS2 Range Checker Configuration . . . 169
Table 90 Other VCAP IS2 Key Configurations . . . 170
Table 91 Combining VCAP IS2 Actions . . . 171
Table 92 VCAP IS2 Statistics and Diagnostics . . . 172
Table 93 ANA_ACL Address Swapping and Rewriting Registers . . . 173
Table 94 ANA_ACL Routing Registers . . . 174
Table 95 ANA_ACL PTP Processing and Rewriting Registers . . . 175
Table 96 MAC Table Entry . . . 177
Table 97 MAC Table Access Registers . . . 178
Table 98 MAC Table Access Commands . . . 180
Table 99 Scan Filters . . . 183
Table 100 Scan Modification Actions . . . 183
Table 101 Forwarding Registers . . . 184
Table 102 Hardware-Based Learning . . . 186
Table 103 Automated Age Scan Registers Overview . . . 189
Table 104 ISDX Registers Overview . . . 191
Table 105 Analyzer Interrupt Handling Registers Overview . . . 193
Table 106 PGID Registers Overview . . . 194
Table 107 Source Table Registers Overview . . . 196
Table 108 Aggregation Table Registers Overview . . . 196
Table 109 GLAG Forward Registers Overview . . . 197
Table 110 Policer Control Registers Overview . . . 198
Table 111 Service and Bundle Dual Leaky Bucket Table Entries . . . 201
Table 112 BUM Single Leaky Bucket Table Entries . . . 202
Table 113 ACL Policer Table Entries . . . 203

Table 114 Priority Policer Table Entry . . . 204
Table 115 Port Policer Table Entry . . . 204
Table 116 Storm Policer Table Entry . . . 206
Table 117 Analyzer Port Statistics Register Overview . . . 207
Table 118 Analyzer Service Statistics Registers Overview . . . 210
Table 119 Queue Statistics Registers Overview . . . 212
Table 120 Bundle Policer Statistics Registers Overview . . . 213
Table 121 BUM Policer Statistics Registers Overview . . . 214
Table 122 ACL Policer Statistics Registers Overview . . . 215
Table 123 Analyzer Routing Statistics Registers Overview . . . 216
Table 124 Analyzer sFlow Registers Overview . . . 217
Table 125 ANA_AC Mirror Registers Overview . . . 218
Table 126 VOE Terminology . . . 223
Table 127 IFH Settings for Down-VOE Frame Injection . . . 228
Table 128 IFH Settings for Up-VOE Frame Injection . . . 229
Table 129 Port Definitions in Shared Queue System . . . 284
Table 130 Ingress Port Modes . . . 286
Table 131 Analyzer Result . . . 286
Table 132 Buffer Control Registers Overview . . . 287
Table 133 Buffer Status Registers Overview . . . 287
Table 134 Statistics Modes . . . 288
Table 135 Forward Pressure Configuration Registers Overview . . . 288
Table 136 Analyzer Destination Selection . . . 289
Table 137 Frame Forwarder Registers Overview . . . 289
Table 138 Analyzer Congestion Results . . . 290
Table 139 Strict Priority Sharing Configuration Settings . . . 292
Table 140 Per Priority Sharing Configuration Settings . . . 293
Table 141 WRED Sharing Configuration . . . 294
Table 142 Threshold Configuration Overview . . . 294
Table 143 Congestion Control Registers Overview . . . 295
Table 144 Frame Forwarder Monitoring Registers Overview . . . 295
Table 145 Queue Mapping Registers Overview . . . 299
Table 146 Queue Limitation Conditions . . . 300
Table 147 Queue Limitation Registers . . . 300
Table 148 Shaper Registers Overview . . . 303
Table 149 Scheduler Arbitration Registers Overview . . . 304
Table 150 Miscellaneous Scheduler Registers Overview . . . 304
Table 151 Frame Aging Registers Overview . . . 305
Table 152 Statistics Address Mapping . . . 306
Table 153 Statistics Registers Overview . . . 306
Table 154 EEE Registers Overview . . . 307
Table 155 Calendar Registers Overview . . . 308
Table 156 Frame Table Entry Register Overview (2048 entries) . . . 310
Table 157 DTI Table Entry Registers Overview . . . 311
Table 158 Miscellaneous DTI Parameters . . . 311
Table 159 TTI Timer Ticks Registers Overview . . . 315
Table 160 TTI Parameters (2048 entries) . . . 315
Table 161 TTI Calendar Registers Overview . . . 316
Table 162 Miscellaneous TTI Registers Overview . . . 317
Table 163 AFI TUPE Registers Overview . . . 317
Table 164 Port Parameters . . . 322
Table 165 ES0 Configuration Registers . . . 323
Table 166 VCAP_ES0 Keys . . . 324
Table 167 VCAP_ES0 Actions . . . 325
Table 168 Rewriter Pipeline Point Definitions . . . 331
Table 169 Pipeline Actions . . . 332
Table 170 Pipeline Point Updating Rules . . . 332
Table 171 Frame Loopback Options . . . 333
Table 172 Pipeline Action Updates . . . 333

Table 173 Loopback IFH Updates . . . 334
Table 174 Egress Statistics Control Registers . . . 334
Table 175 ESDX Index Selection . . . 335
Table 176 Egress Statistics Update . . . 336
Table 177 Mapping Table Configuration Registers . . . 336
Table 178 Port VLAN Tag Configuration Registers . . . 338
Table 179 ES0 VLAN Tag Actions . . . 339
Table 180 Port-Based DSCP Editing Registers . . . 343
Table 181 Rewriter VStaX Configuration Registers . . . 343
Table 182 GCPU Configuration Registers . . . 345
Table 183 Routing SMAC Configuration Registers . . . 346
Table 184 L3 Routing Egress Mapping VLAN (EVMID) . . . 346
Table 185 Rewriter OAM Outer TCI Registers . . . 347
Table 186 Y.1731 OAM MIP Registers . . . 348
Table 187 ISDX Table Configuration Registers . . . 350
Table 188 Mirror Probe Configuration Registers . . . 351
Table 189 MPLS Encapsulation Configuration Registers . . . 353
Table 190 IFH Configuration Registers . . . 358
Table 191 Replication Terminology . . . 359
Table 192 Basic Port Setup Configuration Registers Overview . . . 360
Table 193 Buffer Maintenance Configuration Register Overview . . . 360
Table 194 PAUSE Frame Generation Configuration Registers Overview . . . 360
Table 195 PAUSE Frame Reception Configuration Registers Overview . . . 361
Table 196 PFC PAUSE Frame Generation Configuration Registers Overview . . . 361
Table 197 Half-Duplex Mode Flow Control Configuration Register Overview . . . 361
Table 198 Aging Configuration Register Overview . . . 362
Table 199 Aging Status Register Overview . . . 362
Table 200 Rate Limiting Common Configuration Registers Overview . . . 362
Table 201 Rate Limiting Frame Overhead Configuration Registers Overview . . . 363
Table 202 Rate Limiting Payload Data Rate Configuration Registers Overview . . . 363
Table 203 Rate Limiting Frame Rate Registers Overview . . . 364
Table 204 Error Detection Status Registers Overview . . . 364
Table 205 Layer 1 Timing Configuration Registers . . . 364
Table 206 Layer 1 Timing Recovered Clock Pins . . . 365
Table 207 Recovered Clock Settings for 1 Gbps or Lower . . . 365
Table 208 Recovered Clock Settings for 2.5 Gbps . . . 366
Table 209 Recovered Clock Settings for PLL . . . 366
Table 210 Squelch Configuration for Sources . . . 367
Table 211 Timing Parameters . . . 368
Table 212 Rewriter PTP Configuration . . . 369
Table 213 Rewriter Operations for One-Step Operation . . . 370
Table 214 LoadStore Controller . . . 373
Table 215 IEEE 1588 Configuration Registers Overview . . . 373
Table 216 PTP Actions in Rewriter . . . 374
Table 217 Rewriter Registers . . . 374
Table 218 Port Module Registers . . . 375
Table 219 I/O Delays at 156.25 MHz . . . 375
Table 220 I/O Delays at 52 MHz . . . 375
Table 221 VRAP Registers . . . 376
Table 222 VLAN Table Configuration Before APS . . . 385
Table 223 VLAN TUPE Command for Ring Protection Switching . . . 386
Table 224 VLAN Table Configuration After APS . . . 386
Table 225 MAC Table Flush, First Command . . . 387
Table 226 ISDX Values for E-Line Linear Protection . . . 387
Table 227 E-Line Linear Protection Switching Configuration . . . 388
Table 228 MC_IDX MAC-Based Learning Configuration . . . 388
Table 229 VLAN TUPE Command for E-Line Linear Protection Switching . . . 389
Table 230 Strapping . . . 391
Table 231 Clocking and Reset Configuration Registers . . . 393

Table 232 Shared Bus Configuration Registers . . . 396
Table 233 SI Slave Mode Pins . . . 400
Table 234 MIIM Slave Pins . . . 402
Table 235 MIIM Registers . . . 402
Table 236 VCore Shared Bus Access Registers . . . 403
Table 237 Mailbox and Semaphore Registers . . . 404
Table 238 Manual PCIe Bring-Up Registers . . . 406
Table 239 Base Address Registers . . . 408
Table 240 PCIe Outbound Interrupt Registers . . . 408
Table 241 Outbound Access Registers . . . 409
Table 242 PCIe Access Header Fields . . . 409
Table 243 FDMA PCIe Access Header Fields . . . 410
Table 244 Power Management Registers . . . 411
Table 245 PCIe Wake Pin . . . 411
Table 246 FDMA Registers . . . 412
Table 247 SI Boot Controller Configuration Registers . . . 420
Table 248 Serial Interface Pins . . . 421
Table 249 SI Master Controller Configuration Registers Overview . . . 423
Table 250 SI Master Controller Pins . . . 424
Table 251 DDR3/DDR3L Controller Registers . . . 427
Table 252 General DDR3/DDR3L Timing and Mode Parameters . . . 428
Table 253 Memory Controller Configuration . . . 428
Table 254 Timer Registers . . . 432
Table 255 UART Registers . . . 433
Table 256 UART Interface Pins . . . 433
Table 257 Two-Wire Serial Interface Registers . . . 434
Table 258 Two-Wire Serial Interface Pins . . . 435
Table 259 MIIM Registers . . . 437
Table 260 MIIM Management Controller Pins . . . 437
Table 261 GPIO Registers . . . 439
Table 262 GPIO Overlaid Functions . . . 440
Table 263 Parallel Signal Detect Pins . . . 441
Table 264 SIO Registers . . . 442
Table 265 SIO Controller Pins . . . 442
Table 266 Blink Modes . . . 445
Table 267 SIO Controller Port Mapping . . . 445
Table 268 Fan Controller . . . 447
Table 269 Fan Controller Pins . . . 447
Table 270 Temperature Sensor Registers . . . 448
Table 271 Integrity Monitor Registers . . . 449
Table 272 Memories with Integrity Support . . . 451
Table 273 Interrupt Controller Registers . . . 452
Table 274 Interrupt Sources . . . 452
Table 275 Interrupt Destinations . . . 454
Table 276 External Interrupt Pins . . . 456
Table 277 Reference Clock Inputs . . . 458
Table 278 PLL Clock Output . . . 458
Table 279 DDR3 SDRAM Signals . . . 459
Table 280 DDR3L SDRAM Signals . . . 459
Table 281 1G Transmitter . . . 460
Table 282 1G Receiver . . . 460
Table 283 6G Transmitter . . . 460
Table 284 6G Receiver . . . 461
Table 285 I/O Signals . . . 461
Table 286 GPIO, SI, JTAG, and Miscellaneous Signals DC Specifications . . . 462
Table 287 Thermal Diode Parameters . . . 462
Table 288 REFCLK Reference Clock . . . 463
Table 289 REFCLK2 Reference Clock . . . 463
Table 290 PLL Clock Outputs . . . 465

Table 291 100BASE-FX, SGMII, SFP, 1000BASE-KX Transmitter . . . 465
Table 292 100BASE-FX, SGMII, SFP, 1000BASE-KX Receiver . . . 465
Table 293 100BASE-FX, SGMII, SFP, 2.5G, 1000BASE-KX Transmitter . . . 466
Table 294 100BASE-FX, SGMII, SFP, 2.5G, 1000BASE-KX Receiver . . . 467
Table 295 PCIe Transmitter . . . 468
Table 296 PCIe Receiver . . . 468
Table 297 nRESET Timing Specifications . . . 469
Table 298 MIIM Timing Specifications . . . 470
Table 299 SI Boot Timing Specifications for Master Mode . . . 470
Table 300 SI Timing Specifications for Master Mode . . . 471
Table 301 SI Timing Specifications for Slave Mode . . . 472
Table 302 DDR3/DDR3L SDRAM Input Signal AC Characteristics . . . 473
Table 303 DDR3/DDR3L SDRAM Output Signal AC Characteristics . . . 474
Table 304 JTAG Interface AC Specifications . . . 476
Table 305 Serial I/O Timing Specifications . . . 477
Table 306 Recovered Clock Output AC Specifications . . . 478
Table 307 Two-Wire Serial Interface AC Specifications . . . 479
Table 308 IEEE1588 Time Tick Output AC Specifications . . . 480
Table 309 Operating Current . . . 480
Table 310 Power Consumption . . . 481
Table 311 Power Consumption, Reduced Configuration . . . 481
Table 312 Recommended Operating Conditions . . . 481
Table 313 Stress Ratings . . . 482
Table 314 Pin Type Symbol Definitions . . . 484
Table 315 DDR3/DDR3L SDRAM Pins . . . 485
Table 316 GPIO Pins . . . 486
Table 317 JTAG Interface Pins . . . 486
Table 318 MII Management Interface Pins . . . 486
Table 319 Miscellaneous Pins . . . 487
Table 320 PCI Express Interface Pins . . . 487
Table 321 Power Supply and Ground Pins . . . 488
Table 322 SERDES1G Pins . . . 488
Table 323 SERDES6G Pins . . . 488
Table 324 Serial CPU Interface Pins . . . 489
Table 325 System Clock Interface Pins . . . 489
Table 326 Twisted Pair Interface Pins . . . 490
Table 327 Thermal Resistances . . . 500
Table 328 Recommended Skew Budget . . . 504
Table 329 Ordering Information . . . 508

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Revision 4.1

Revision 4.1 was published in May 2019. The following is a summary of changes in revision 4.1 of this document.

- VeriPHY™ Cable Diagnostics section was updated. For more information, see VeriPHY™ Cable Diagnostics, page 43.
- The Design Consideration chapter was updated for OAM statistics. For more information, see Design Considerations, page 507.
- The Pins by Function section was updated with proper heading style.
- The VeriPHY Control 1/2/3 sections were removed.

1.2 Revision 4.0

Revision 4.0 was published in September 2017. It was the first production-level publication of this document.

2 Product Overview

The VSC7430 Carrier Ethernet switch targets IP Edge demarcation and access equipment to deliver Enterprise and Mobile Backhaul. VSC7430 is based on Virtualized Service Aware Architecture (ViSAA™), a silicon implementation that offers an unmatched level of Service Edge and Carrier Ethernet (CE) networking features. ViSAA achieves wirespeed performance for even the most feature-rich Carrier Ethernet services. VSC7430 also integrates VeriTime™, Microsemi's patent-pending timing technology that delivers the industry's most accurate IEEE 1588v2 timing implementation to meet today's LTE/LTE-A requirements.

The VSC7430 switch contains four 10/100/1000 Mbps ports (up to two copper ports), and two 10/100/1000/2500 Mbps ports. VSC7430 provides a rich set of Carrier Ethernet switching features such as hierarchical QoS, hardware-based OAM (Ethernet and MPLS/MPLS-TP) and SAM, protection switching, and Synchronous Ethernet. Using provider bridging (Q-in-Q) and MPLS/MPLS-TP technology, VSC7430 delivers MEF CE 2.0 EVCs. The Versatile Content Aware Processor (VCAP™) architecture is scalable and flexible for secure applications. It features advanced TCAM classification in both ingress and egress. Per-EVC features include a rich set of statistics, OAM for end-to-end performance monitoring, and dual-rate policing and shaping.

IPv4/IPv6 Layer 3 (L3) routing and IPv4/IPv6 L3 multicast groups are supported. L3 security features include source guard and unicast Reverse path Forwarding (uRFP) tasks.

The device contains a powerful 500 MHz MIPS-24KEc CPU enabling full management of the switch and advanced Carrier Ethernet applications.

The VSC7430 device is a 9 Gbps Carrier Ethernet switch with up to six ports supporting the following main port configurations:

- 4 × 1G SGMII ports + 2 × 2.5G SGMII ports
- 6 × 1G SGMII ports
- Two of the 1G SGMII ports are dual-media ports with a choice of 1G SGMII or 1G CuPHY

In addition, the device supports one 10/100/1000 Mbps SGMII/SerDes Node Processor Interface (NPI) Ethernet port.

2.1 General Features

This section lists the key device features and benefits.

2.1.1 Layer 2 and Layer 3 Forwarding

- IEEE802.1Q switch with 4K VLANs and 8K MAC table entries
- Push/pop/translate up to three VLAN tags on ingress and egress
- RSTP and MSTP support
- Fully non-blocking wire-speed switching performance for all frame sizes
- IPv4/IPv6 unicast and multicast Layer 2 switching with up to 8K groups and 128 port masks
- IPv4/IPv6 unicast and multicast Layer 3 forwarding (routing) with reverse path forwarding (RPF) support
- IGMPv2, IGMPv3, MLDv1, and MLDv2 support

2.1.2 Carrier Ethernet Support

- MEF CE 2.0 ready: Eight Class of Service (CoS) Ethernet Virtual Connections (EVC) per-EVC OAM, dual-rate policing and shaping, queuing, and statistics
- Ethernet forwarding based on up to 8K MAC table or service classification, using up to three VLAN tags (Q-Q-Q)
- Ethernet pseudowire over MPLS, and MPLS LSR forwarding
- Reserved label and ACH processing for MPLS OAM
- One OAM Down-MEP per port plus a pool of OAM MEPs supporting Up-MEP and Down-MEP functions
- Service Activation Test frame generation and checking

2.1.3 Timing and Synchronization

- Synchronous Ethernet, with four clock outputs recovered from any port
- IEEE 1588-2008 (v2) with nanosecond-accurate time stamping for one-step and two-step clocks
- Hardware processing and PTP frame generation

2.1.4 Quality of Service (QoS)

- One megabytes of integrated shared packet memory
- Eight QoS classes per port
- TCAM-based classification with pattern matching against Layer 2 through Layer 4 information
- Bandwidth guarantees per EVC and COS obtained through flexible configuration of scheduling elements providing Hierarchical QoS scheduling and dual-rate shaping
- Dual-rate service policers, dual-rate service bundle policers, eight single-rate priority policers per port, and two single-rate port policers for each port
- Flexible 2K ingress QoS mappings and 4K egress QoS mappings for VLAN tags, DSCP values, and MPLS labels.
- Up to 512 egress VLAN tag and MPLS label operations
- Priority-based flow control (PFC) (IEEE 802.1Qbb)
- Audio/Video bridging (AVB) with support for time-synchronized, low-latency audio and video streaming services

2.1.5 Security

- Versatile Content Aware Processor (VCAP™) packet filtering engine using ACLs for ingress and egress packet inspection
- Storm controllers for flooded broadcast, flooded multicast, and flooded unicast traffic, selectable per service
- Per-port, per-address registration for copying/redirecting/discarding
- 32 VCAP single-rate policers

2.1.6 Management

- VCore-III™ CPU system with integrated 500 MHz MIPS 24KEc CPU with MMU and DDR3/DDR3L SDRAM controller
- PCIe 1.x CPU interface
- CPU frame extraction (eight queues) and injection (two queues) through DMA, which enables efficient data transfer between Ethernet ports and CPU/PCIe
- EJTAG debug interface
- Configurable 16-bit or 8-bit DDR3 SDRAM interface supporting up to one gigabyte (GB) of memory
- Thirty-seven, pin-shared general-purpose I/Os
  - Serial GPIO and LED controller controlling up to 32 ports with four LEDs each
  - Triple PHY management controller (MIIM)
  - Dual UART
  - Dual built-in two wire serial interface multiplexer
  - External interrupts
  - 1588 synchronization I/Os
  - SFP loss of signal inputs
- External access to registers through PCIe, SPI, MIIM, or through an Ethernet port with inline Versatile Register Access Protocol (VRAP)
- Per-port counter set with support for the RMON statistics group (RFC 2819) and SNMP interfaces group (RFC 2863)
- Energy Efficient Ethernet (EEE) (IEEE 802.3az)
- Synchronous Ethernet, with four clock outputs recovered from any port
- Support for CPU modes with internal CPU only, external CPU only, or dual CPU
- Carrier Ethernet software API for service creation and management

2.1.7 Product Parameters

All SerDes, copper transceivers, packet memory, and configuration tables necessary to support network applications are integrated. The following table lists the primary parameters for the device.

Table 1 • **Product Parameters**

Port Configurations	
Maximum bandwidth excluding 1G NPI port	9 Gbps
Maximum number of ports excluding 1G NPI port	6
10G SFI ports (also capable of 2.5G or 1G SGMII)	0
2.5G SGMII ports (also capable of 1G SGMII)	2
1G SGMII ports excluding 1G NPI port	2
1G dual media ports (1G SGMII or 1G CuPHY)	2
1G SGMII NPI port	1
Layer 2 Switching	
Packet buffer	8 Mb
MAC table size	8K
Layer 2 multicast port masks	128
Super VCAP blocks (1K x 36 bits per block)	6
VCAP CLM entries (36 bits) per super VCAP block	1K
VCAP LPM entries (36 bits) per super VCAP block	1K
VCAP IS2 entries (288 bits) per super VCAP block	128
VCAP ES0 entries	512
Layer 3 Routing	
Router legs	32
IP unicast routes/hosts per super VCAP block allocated to VCAP LPM for unicast routing	IPv4: 1K IPv6: 256
Next-hop/ARP table entries	256
IP (S,G) or (*, G) multicast groups per super VCAP block allocated to VCAP LPM for multicast routing	IPv4: 512 IPv6: 128
Multicast router leg masks	256
ECMPs	16
Quality of Service and Security	
Ingress QoS mapping table entries	2K
Egress QoS mapping table entries	4K
Security enforcement (ACL) rules (288 bits) per super VCAP block allocated to VCAP IS2	128
Carrier Ethernet	
Bi-directional Carrier Ethernet connections (E-Lines, MPLS pseudo-wires, or MPLS LSPs)	4 COS: 128 8 COS: 64
EVC bidirectional endpoints (ISDX and ESDX)	512
DLB policers	512
ISDX statistics	1K
ESDX statistics	1K

Table 1 • **Product Parameters *(continued)***

Hardware MEPs in addition to 1/port for port OAM	192
Hardware MIPs	128
MPLS encapsulations (tunnel LSPs and Ethernet headers)	128

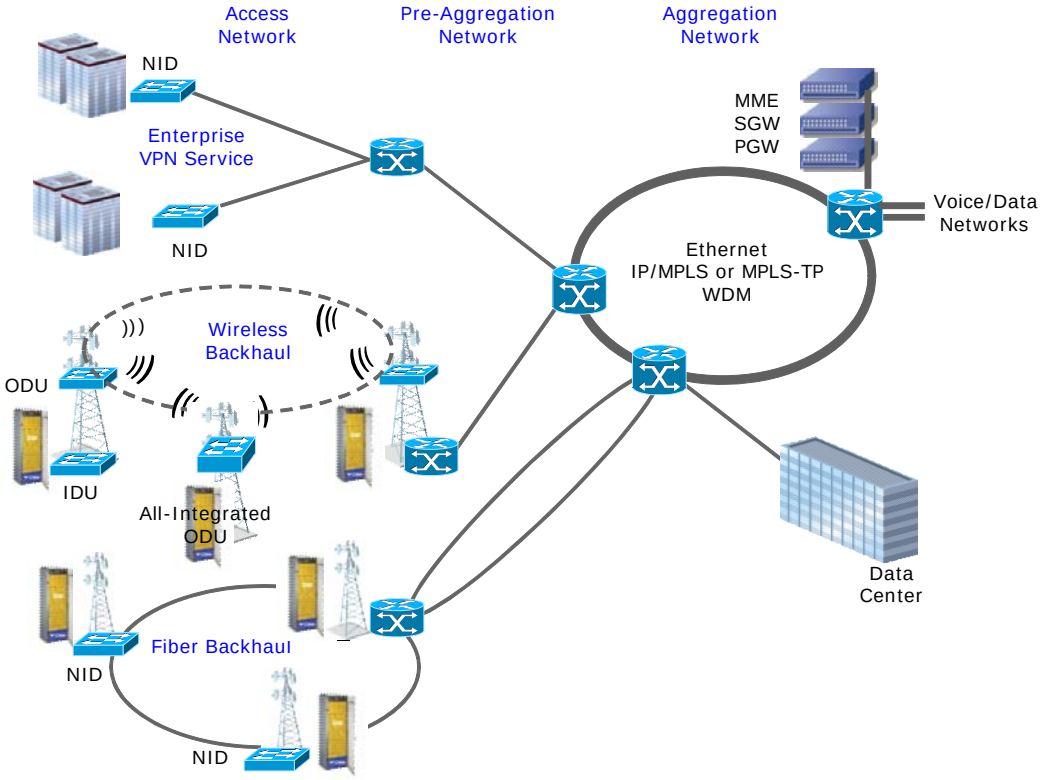
2.2 Applications

The device targets the following applications.

- Small cell backhaul (microwave and millimeter wave, for example) equipment such as split-mount Outdoor Unit (ODU), Indoor Unit (IDU), and all-integrated ODU.
- Network Interface Devices (NIDs) targeting delivery of MEF CE2.0 Carrier Ethernet services for mobile backhaul and Enterprise VPNs. Carrier Ethernet networks may be based on Provider-Bridged Ethernet and MPLS-TP.
- Small Cell equipment (micro, femto, and pico) targeting delivery of 4G/LTE mobile services. This small cell equipment integrates radio front-end and networking components.

The following illustration shows these applications on a single-carrier network with fixed/mobile convergence.

Figure 1 • **Converged Access and Aggregation Network Application**



2.2.1 Wireless Backhaul

Wireless backhaul equipment is available with the following configuration options:

- Traditional split-mount with separate Outdoor Units (ODUs) and Indoor Units (IDUs). In the traditional split, the modem functionality exists in the IDU, and the interface between the ODU and IDU is a modulated waveform.
- Ethernet split-mount also with separate ODU and IDU. In this configuration, the modem functionality exists in the ODU along with some packet switching functions such as redundancy protection. The interface between the ODU and IDU is Ethernet packets.

- All-integrated ODU, where the full ODU and IDU functionality is integrated into one piece of outdoor equipment.

The device targets all of these configurations. Networking features enable wireless backhaul topologies such as chains, trees, and rings. Networking features are especially important in wireless backhaul, which can result in relatively arbitrary daisy-chaining in the backhaul. Typically, smaller switches are present in ODUs and larger switches in IDUs, which often act as aggregators or localized switching centers.

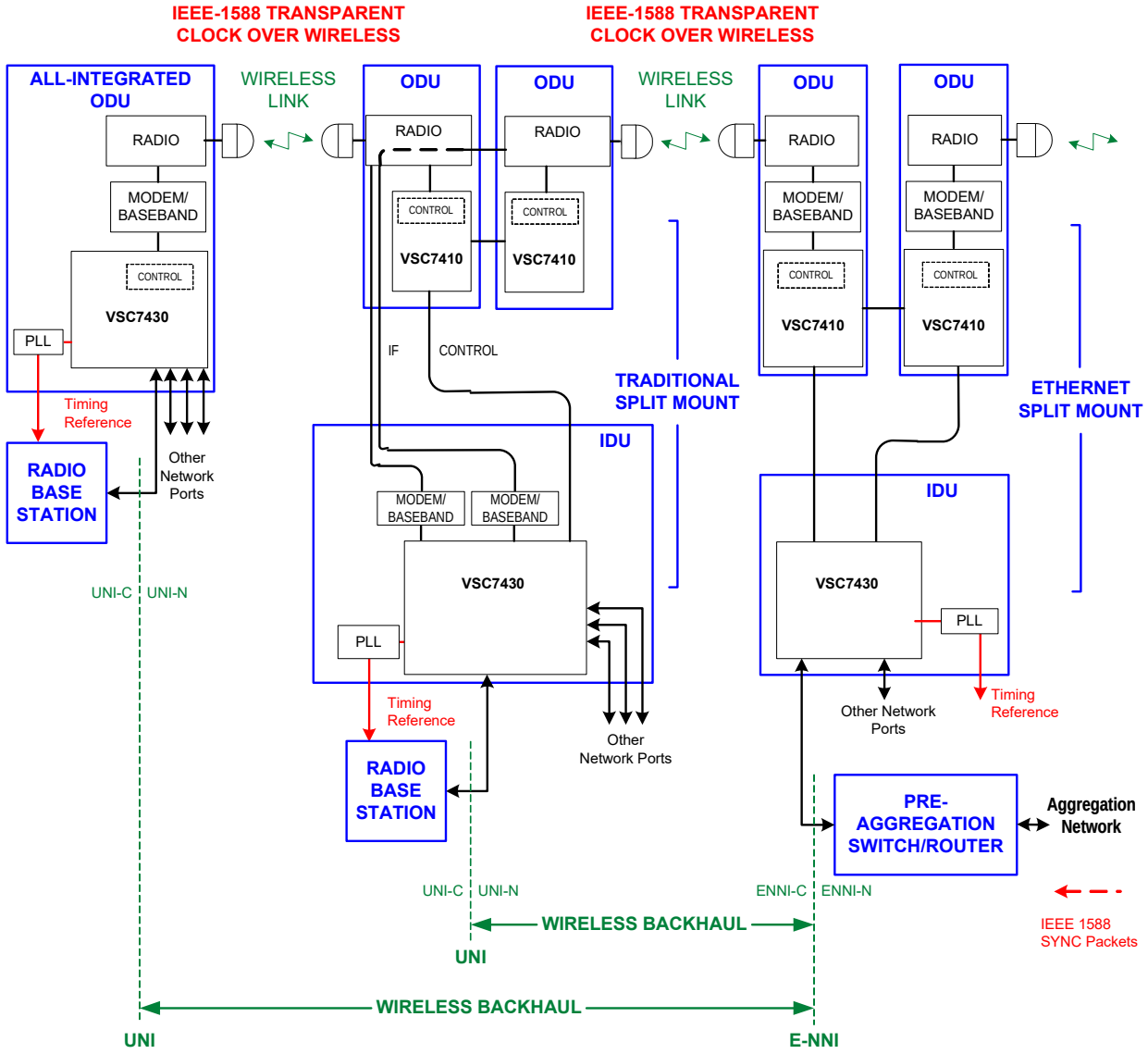
The following illustration shows an all-integrated ODU on the left, a traditional split- mount ODU/IDU in the middle, and an Ethernet split-mount ODU/IDU on the right. In some installations, the Radio Base Station and the all-integrated ODU functionality can also be built into a single piece of equipment.

In the illustration, the full backhaul is made up of the wireless backhaul section plus the wired backhaul section, operated by two separate network providers. Only the wireless backhaul section is illustrated. Other models of network ownership are possible.

The wireless backhaul network provides a transport service between each Radio Base Station and the Pre-aggregation network. The wireless backhaul operator provides an Ethernet connection between each Radio Base Station (Metro Ethernet Forum UNI interface) and the Pre-aggregation Router (Metro Ethernet Forum E-NNI interface).

The application shown is somewhat simplified. In actual deployment, more complex configurations are common, such as equipment redundancy, network redundancy (linear, ring, mesh, link aggregation), and increased wireless capacity through link bonding.

Figure 2 • **Wireless Backhaul Application**



Microwave radio links are highly susceptible to weather and use adaptive modulation techniques to deliver data reliably during bad weather. As a result, the microwave data transfer rate (link speed) can change rapidly, causing jitter problems for IEEE 1588 and Quality of Service (QoS) problems for the networking gear. The link speed is also limited to a few hundred Mbps per microwave link under good weather conditions, so link bundling and load balancing are common. Next-generation ODUs must support timing and synchronization such as Synchronous Ethernet and IEEE 1588 packet based timing.

Small IDUs (pizza boxes) can support Carrier Ethernet features similar to Network Interface Devices (NID). In addition, IDU functionality can include MPLS-TP functionality.

Integrated ODU/IDU equipment also exists, especially for 4G/LTE small cell backhaul. These units have a low port count but include advanced features such as MPLS TP, protection switching mechanisms, and strong QoS.

The following key features are supported for wireless backhaul.

- Low size, low cost, and low power, especially for the ODUs and all-integrated ODU
- Integrated 1G copper PHYs
- Ethernet, MPLS-TP, and IP networking features at the IDUs and all-integrated ODUs
- MEF Service OAM, and Ethernet/MPLS-TP OAM for the NNIs

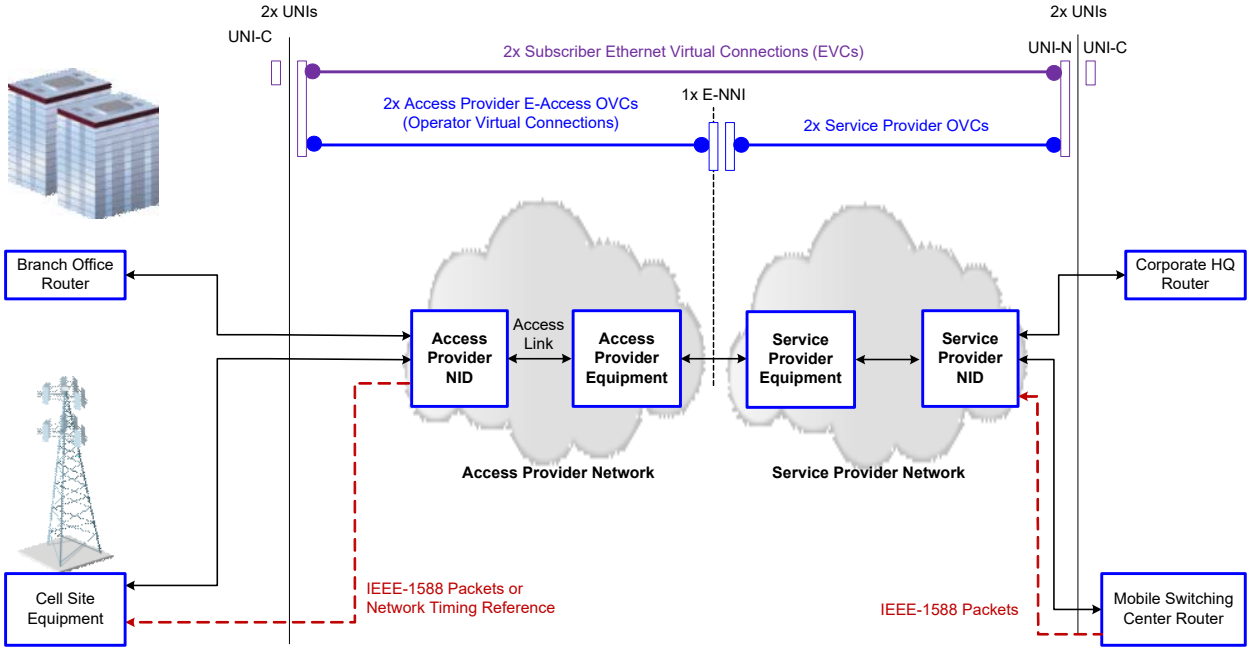
- IEEE 1588 functionality delivering high-precision timing to 4G/LTE Base Stations. Various proprietary techniques are supported to combat the jitter created by adaptive modulation on a wireless link, including ability to provide an IEEE 1588 transparent clock function over a wireless link.
- Strong QoS to handle the varying link speed
  - IEEE 802.1Qbb priority-based flow control can be used as in-band dynamic flow control.
  - Versatile Register Access Protocol (VRAP) may also be used to dynamically adjust scheduler/shaper rates in-band.
  - Large integrated buffer memory, advanced hierarchical scheduling and shaping, and WRED enable efficient bandwidth use without jeopardizing low-latency traffic such as voice.
- Linear and ring protection mechanisms
- –40 °C ambient to 125 °C junction operating temperature

2.2.2 Network Interface Device (NID)

The following illustrations show three ways to construct multi-operator service. In these examples, the access operator cooperates with the service provider to offer service in areas where the service provider has no footprint. These examples provide MEF-compliant Ethernet Virtual Connection (EVC) services, plus carrier-grade reference timing based on Synchronous Ethernet or IEEE 1588.

In the first reference diagram, the access operator NID provides MEF-compliant E-Access service where the subscriber's service frames are tunneled through the access operator's network back to the service provider's network. Service functionality is implemented primarily in the service provider network.

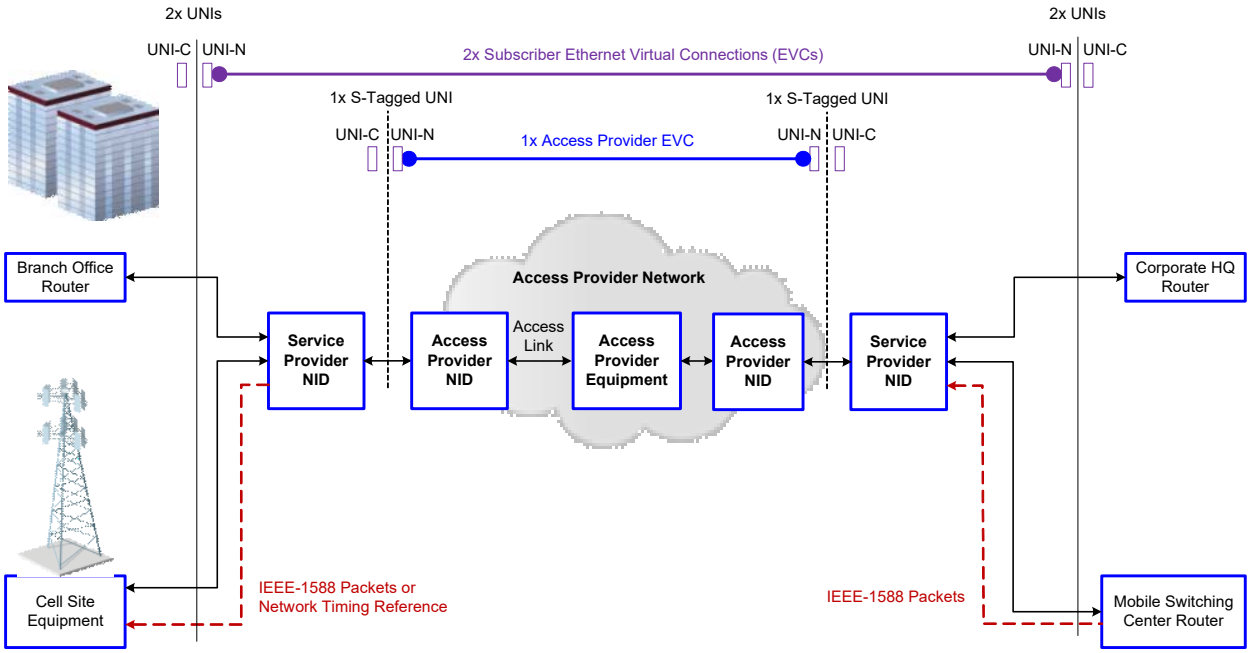
Figure 3 • **E-Access NID Application**



In the following illustration, the service provider locates a NID at the cell site or customer location. Service functionality is therefore implemented primarily in the service provider NID. This example typically results in two NIDs at the edge of the cell sites, or at each customer location. One NID is owned by the access operator, the other is owned by the service provider.

Although not standardized, it is common for the access provider NIDs to offer S-tagged UNI service, which is standard MEF UNI service, but using S-tags to identify each EVC.

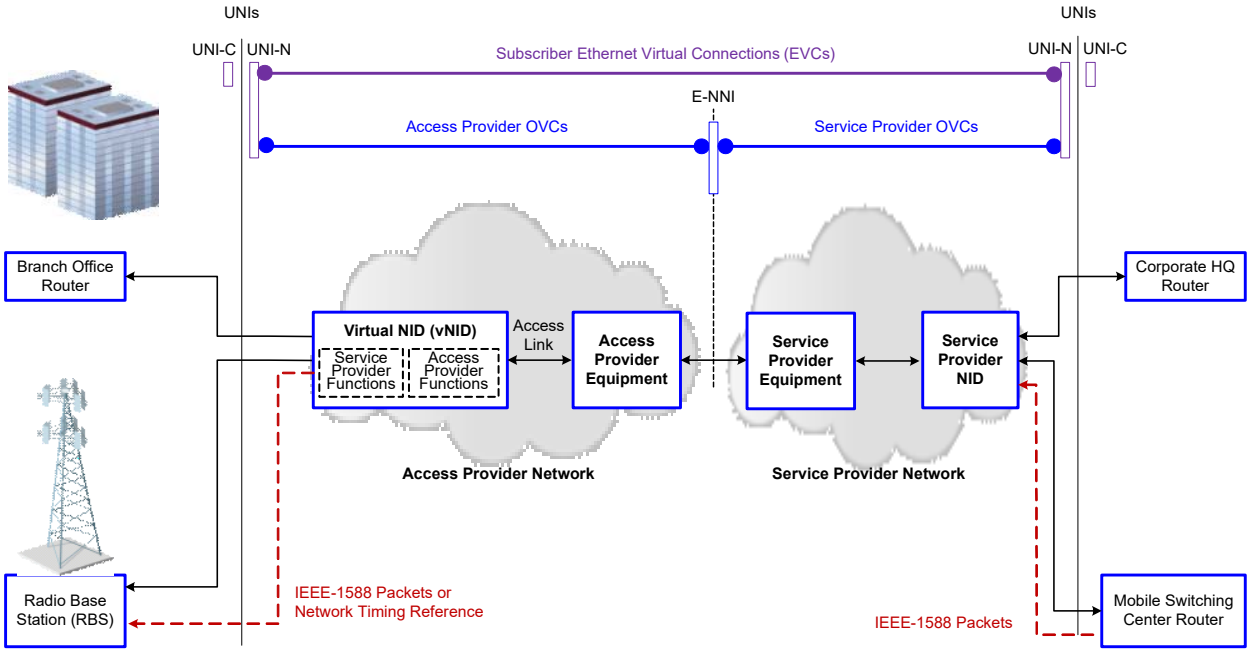
Figure 4 • **S-Tagged NID Application for Multi-Operator Access**



In the following illustration, the access provider NID is upgraded to a virtual NID (vNID), which allows for both access provider and service provider management and OAM functions in one physical box deployed by the access provider.

Multi-domain OAMs must be implemented to enable separate OAM functions for the access provider and the service provider.

Figure 5 • **Virtual NID Application for Multi-Operator Access**



The device is optimized for delivery of MEF EVCs that are based on the Virtualized Service Aware Architecture (ViSAA™). MEF-compliant features include dual-rate policing, dual-rate shaping, statistics, and hardware-based OAM. Hierarchical QoS supports these features at the EVC and UNI/E-NNI level.

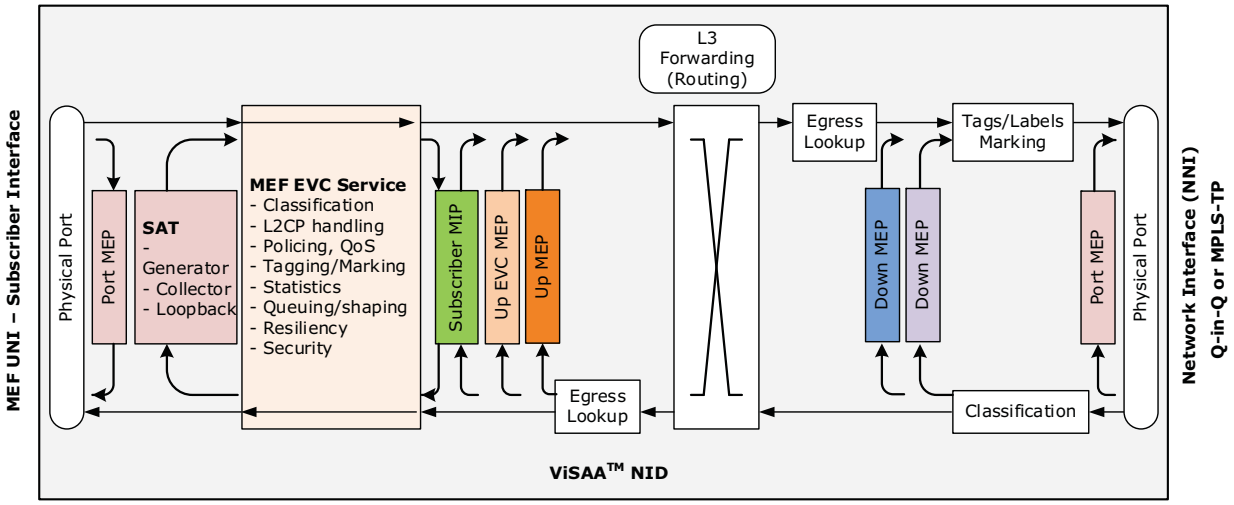
Enterprise VPN services provide connectivity between corporate Headquarters and Branch offices, as well as providing voice, video, internet, storage, and other network-hosted applications.

Mobile backhaul services provide connectivity between radio base stations and mobile switching centers. For mobile backhaul applications, Microsemi offers the industry's highest performance IEEE 1588 solution, uniquely able to deliver LTE Advanced performance for 1588 Slave and one-step transparent clock applications. The device switch family integrates technology for implementing Synchronous Ethernet and IEEE-1588 Equipment Clocks.

Access provider networks predominantly use Provider Bridging (Q-in-Q) and/or MPLS TP technology for backhauling customer traffic further into the network. NIDs generally have a low port count as shown in this example, although it is also possible to have larger NIDs with many customer-facing ports.

The following illustration shows an internal view of a NID implemented using ViSAA™ technology.

Figure 6 • **NID Internal View**



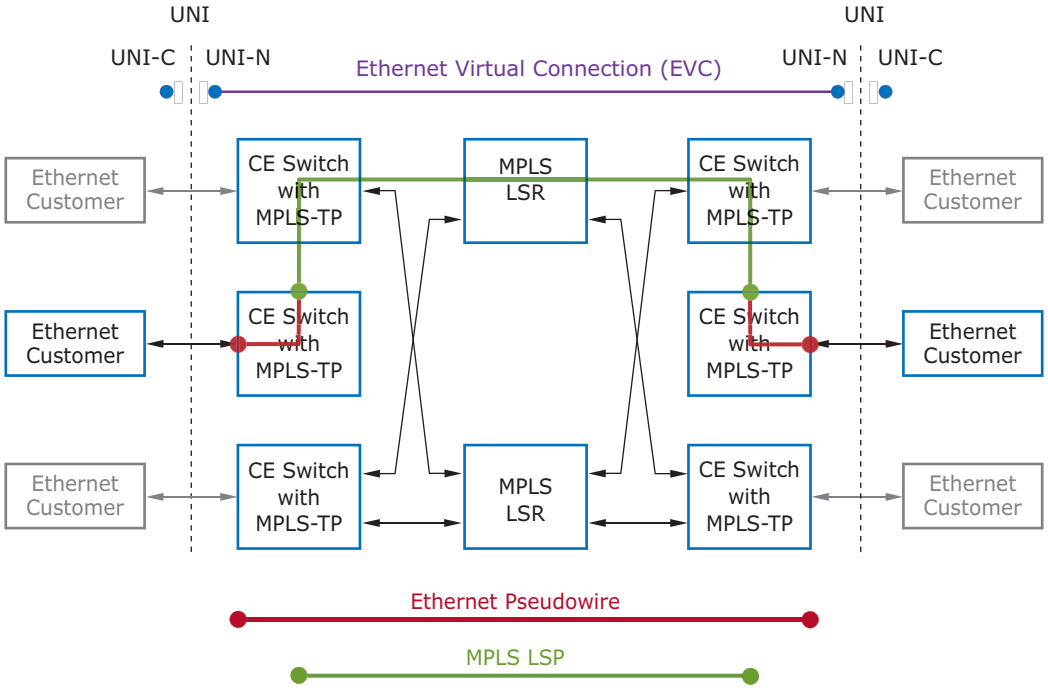
The following key features are supported.

- Highly flexible classification, tagging, and encapsulation for Provider Bridges (Q-in-Q), MPLS pseudowires, and MPLS-TP transport.
- Fully-featured Ethernet-based OAM. Hardware OAM implementation is required for precision and scale.
- At the UNI-C and ENNI, MEF-compliant DLB shaping and Down-MEP functions. Down-MEPs allow the customer to monitor the service purchased from the network (for example, the service provider may monitor the service purchased from the access operator).
- At the UNI-N and ENNI, MEF-compliant dual-rate policing and Up-MEP functions. Up-MEPs allow the Network to monitor the service sold to the customer. These switches support two stacked up MEPs, enabling:
  - One Up MEP to be owned by the service provider while the other Up MEP is owned by the access operator, or
  - One Up MEP to monitor end-to-end Ethernet service as seen by the customer while the other Up MEP monitors internal network connectivity (e.g. Ethernet S-VLAN or MPLS Pseudowire).
- Per-service queuing for customer QoS separation.
- Per-service and color statistics for frame arrivals, departures, and discards.
- Service Activation Testing (SAT) for SLA assurance.
- IEEE-1588 network timing functions. These are especially important for mobile backhaul services, where the NID must provide a high-quality timing reference to the radio base station.
- Linear and ring protection mechanisms as well as path and port protection for dual-homed NID installations.
- Integrated 1G copper PHY ports.
- Routing for network management and control.
- Integrated CPU sub-system for management and control.

2.2.3 Carrier Ethernet Switch with MPLS-TP

The following illustration shows an MEF Ethernet virtual connection (EVC) being offered from an MPLS network. The customer interface (UNI) is Ethernet, and the customer is unaware that the carrier network is using MPLS technologies.

Figure 7 • **Carrier Ethernet Switch with MPLS-TP Application**



The Carrier Ethernet Switches initiating and terminating the pseudowire implement Provider Edge (PE) functions, providing Ethernet-to-MPLS and MPLS-to-Ethernet operations. These are also known as Pseudowire Label Edge Router (LER) operations. The PE switches also initiate and terminate the MPLS Label Switched Path (LSP) connection.

The Carrier Ethernet switches that forward MPLS packets using MPLS label switching provide MPLS-to-MPLS operations. These switches are known as Label Switch Routers (LSRs).

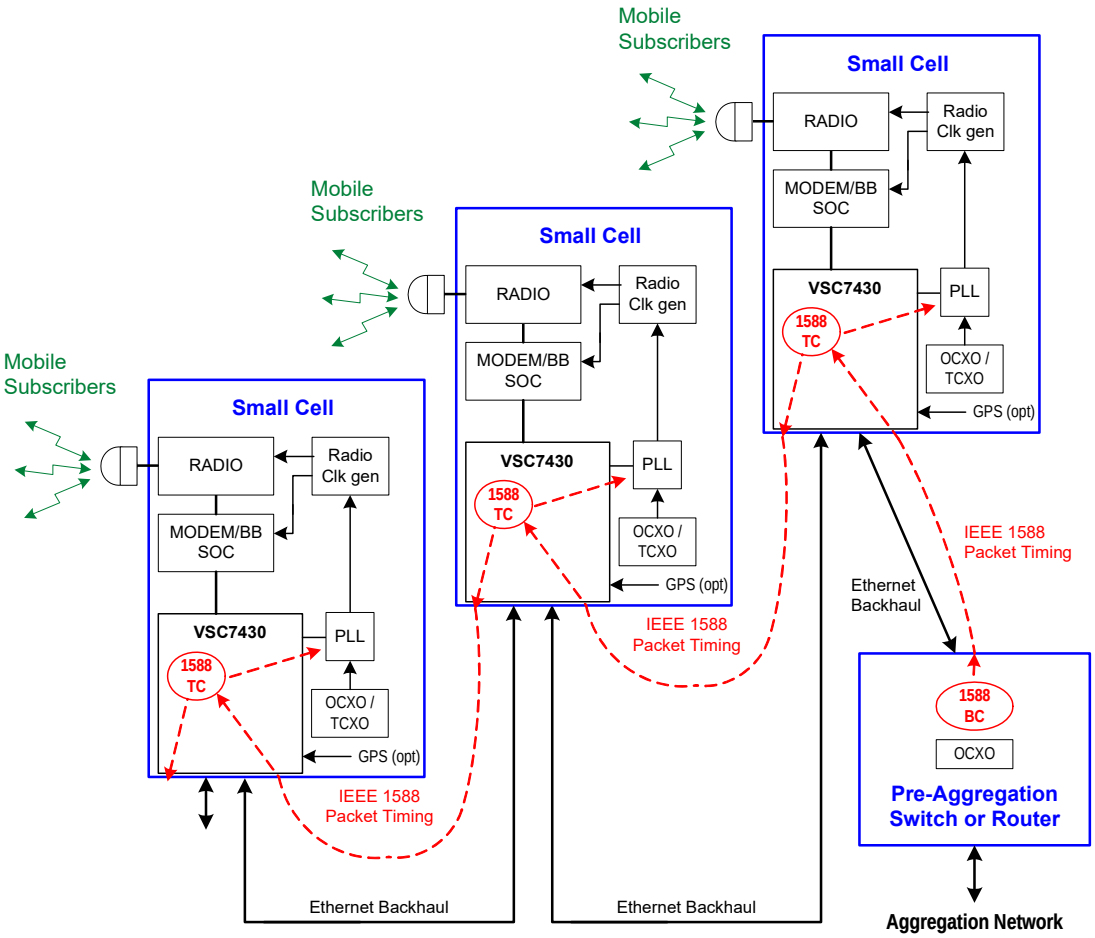
The switch supports these pseudowire LER and MPLS LSR operations and provide the following capabilities:

- EoMPLS PW LER (Single-Segment and Multi-Segment Pseudowire) MPLS LSR
- Hierarchical VPLS (H-VPLS) edge functions (MTU-s)
- MEF-compliant service delivery with per-EVC dual-rate policing and arrival/departure/discard statistics kept for each EVC and each PW
- Concurrent Ethernet OAM at the EVC Service Layer and MPLS OAM for pseudowires and LSPs
- Pseudowire and label switched path layers hierarchical scheduling and dual-rate shaping
- Protection switching using MPLS and Ethernet layers
- IEEE 1588 network timing functions, including transparent clock hardware for PTP delivered over MPLS and pseudowire
- Integrated 1G copper PHY ports
- Integrated CPU sub-system for management and control

2.2.4 Small Cell Application

The following illustration shows multiple 4G/LTE small cells being backhauled to a pre-aggregation switch or router at the edge of a wired network.

Figure 8 • **4G/LTE Small Cell Application**



For small cell applications, the device provides strong backhaul networking along with integrated timing functions, as listed below:

- MEF-compliant Carrier Ethernet backhaul with hardware-based OAM and protection switching capabilities.
- Integrated 1G copper PHY ports.
- Routing for network management and control.
- Small cell timing based on Synchronous Ethernet, IEEE-1588 packets, or optional local GPS receiver.

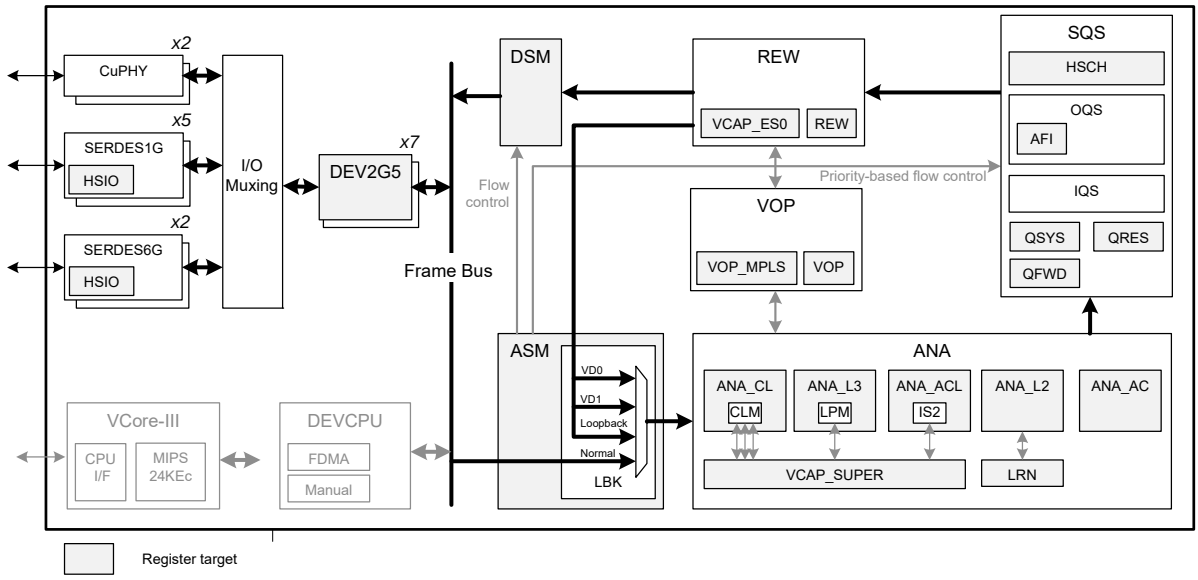
3 Functional Descriptions

This section describes the functional aspects of the VSC7430 device, including available configurations, operational features, and testing functionality. It also defines the device setup parameters that configure the device for a particular application.

The functional descriptions in this section contain some information shared across multiple switch families. As a result, some functional descriptions do not apply to the device. Stacking is not supported, and any references to stacking, stacking links, and learn-all frames should be ignored. Note that even though stacking is not supported, the VStaX header is used by the switch as an internal frame header and communication with external CPUs. DEV10G ports are not supported on the device, and any references to them should also be ignored.

The following illustration shows an RTL block diagram of the physical blocks in the device and how the blocks interconnect. The functional aspects of each block are provided in following sections. The grayed-out blocks represent the VCore-III and DEVCPU blocks. For more information about the VCore-III and DEVCPU blocks, see VCore-III System and CPU Interfaces, page 392.

Figure 9 • **RTL Block Diagram**



3.1 Register Notations

This datasheet uses the following general register notations.

<TARGET>:<REGISTER_GROUP>:<REGISTER>.<FIELD>

<REGISTER_GROUP> is not always present. In that case the following notation is used:

<TARGET>::<REGISTER>.<FIELD>

When a register group does exist, it is always prepended with a target in the notation.

In sections where only one register is discussed or the target (and register group) is known from the context, the <TARGET>:<REGISTER_GROUP>: may be omitted for brevity leading to the following notation:

<REGISTER>.<FIELD>

When a register contains only one field, the .<FIELD> is not included in the notation.

When referring to a specific instance of a register group, specific register instance, or a specific bit in a field, square brackets are used, for example:

<TARGET>:<REGISTER_GROUP>[<register group instance>]:<REGISTER>.<FIELD>

<TARGET>:<REGISTER_GROUP>:<REGISTER>[<register instance>].<FIELD>

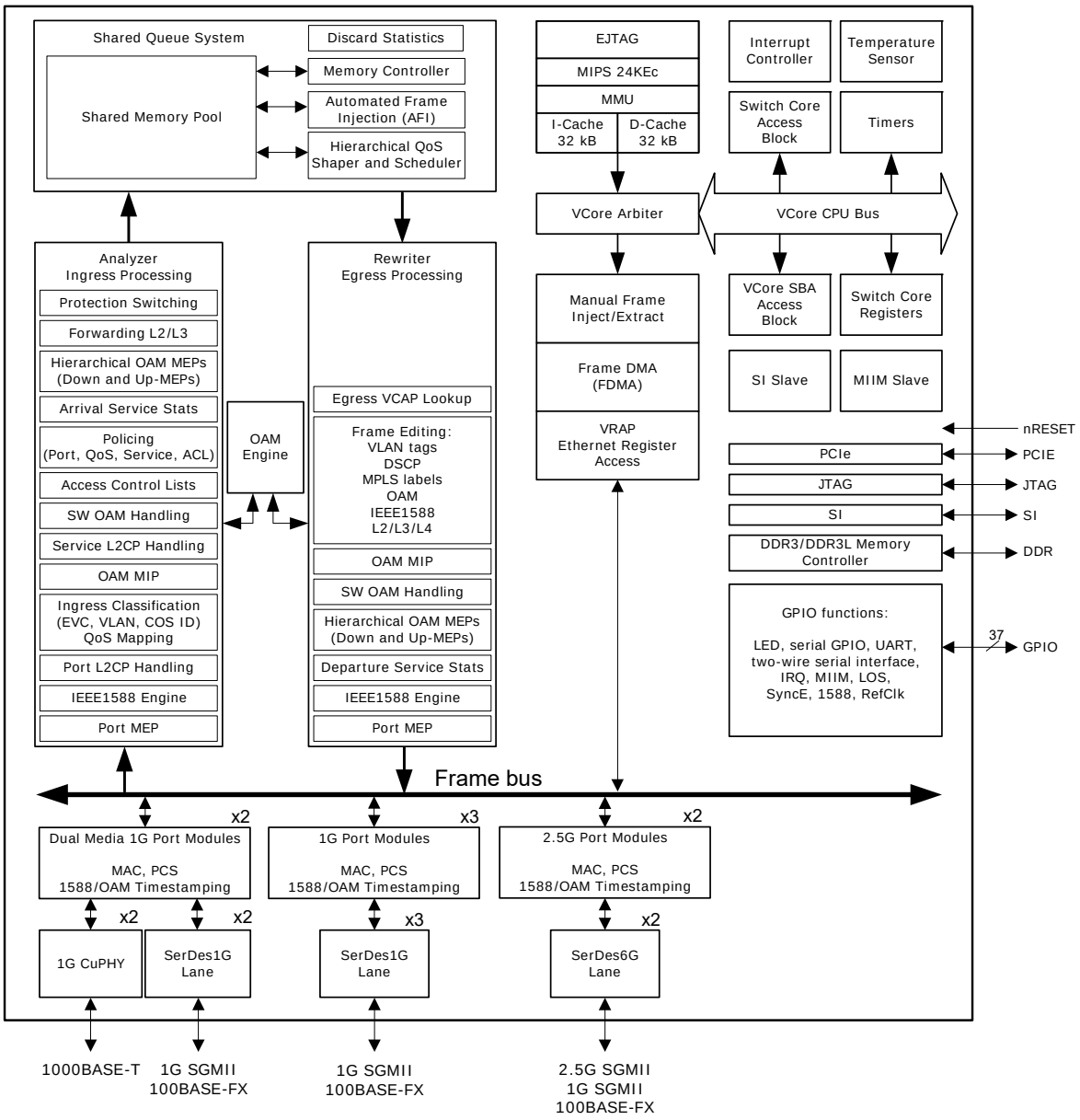
<TARGET>:<REGISTER_GROUP>:<REGISTER>.<FIELD>[<bit number>]

3.2 Functional Overview

This section provides an overview of all the major blocks and functions involved in the forwarding operation, in the same order as a frame traverses through the device. Other major functions of the device, such as the CPU subsystem, OAM processing, and IEEE 1588 PTP processing are also outlined.

The following illustration shows the VSC7430 block diagram.

Figure 10 • **Block Diagram**



3.2.1 Frame Arrival in Ports and Port Modules

The Ethernet interfaces receive incoming frames and forward these to the port modules.

Two ports are dual media ports supporting either a 1G copper PHY or a 1G SGMII SerDes macro. The remaining ports all support SerDes macros.

Each port contains a Media Access Controller (MAC) that performs a full suite of checks, such as VLAN tag-aware frame size checking, frame check sequence (FCS) checking, and pause frame identification.

SerDes ports contain a Physical Coding Sublayer (PCS), which performs 8b/10b or 64b/66b encoding, auto-negotiation of link speed and duplex mode, and monitoring of the link status.

Symmetric and asymmetric pause flow control are both supported as well as priority-based flow control (IEEE 802.1Qbb).

All Ethernet ports support Energy Efficient Ethernet (EEE) according to IEEE 802.3az. The shared queue system is capable of controlling the PCS operating states, which are active or low power. The PCS understands the line signaling as required for EEE. This includes signaling of active, sleep, quiet, refresh, and wake.

3.2.1.1 1G Copper PHY Ports

1G copper PHY ports operate in 1000BASE-T. Auto-negotiation is supported for pause settings and remote fault signaling. Full-duplex is supported for 10/100/1000 Mbps, while half-duplex is supported for 10/100 Mbps.

3.2.1.2 1G SGMII Line Ports

1G SGMII line ports operate in one of the following modes:

- 10/100/1000 Mbps SGMII. The SGMII interface connects to an external copper PHY device or copper SFP optical module with SGMII interface. In this mode, auto-negotiation is supported for link speed, duplex settings, pause settings, and remote fault signaling. Full-duplex is supported for all speeds, while half-duplex is supported for 10/100 Mbps.
- 100BASE-FX. The 100BASE-FX interface connects directly to a 100BASE-FX SFP optical module. Auto-negotiation is not specified for 100BASE-FX and is not supported. Operation is always 100 Mbps and full duplex.
- 1000BASE-X. The 1000BASE-X interface connects directly to a 1000BASE-X SFP optical module. Auto-negotiation is supported for pause settings and remote fault signaling. Operation is always 1000 Mbps and full duplex. 1G backplane Ethernet 1000BASE-KX is fully supported, including auto-negotiation.

3.2.1.3 2.5G SGMII Line Ports

The 2.5G interface connects directly to an SFP optical module. Operation is always 2500 Mbps and full duplex. 2.5G backplane Ethernet is also supported.

The 2.5G SGMII line port can also operate in the modes supported by the 1G SGMII line port.

3.2.2 Basic Classification

Basic frame classification in the ingress processing module (the analyzer) receives all frames before further classification processing is performed. The basic classifier uses configurable logic to classify each frame to a VLAN, QoS class, drop precedence (DP) level, DSCP value, and an aggregation code. This information is carried through the switch together with the frame and affects policing, drop precedence marking, statistics collecting, security enforcement, Layer 2 and Layer 3 forwarding, and rewriting.

The basic classifier also performs a general frame acceptance check. The output from the basic classification can be overwritten or changed by the more intelligent advanced classification using the TCAM-based VCAP.

3.2.3 Virtualized Service Aware Architecture (ViSAA™)

The Microsemi Carrier Ethernet service-aware architecture enables a rich set of Carrier Ethernet service features using multiple lookups into VCAP classification matching (CLM). Up to 64 ingress DSCP

mapping tables (64 entries each) can be used for more efficient TCAM utilization. Carrier Ethernet services include:

- Metro Ethernet Forum (MEF) E-Line, E-LAN, and E-TREE EVCs
- MPLS pseudowires and Label Switched Paths (LSPs)

Carrier Ethernet service policy attributes include:

- Mapping to a service-level dual-rate policer. Multiple services can map to one policer. Up to eight policers can be assigned per service to implement the per-COS per-EVC MEF bandwidth profile.
- Mapping to a VLAN/VSI (MAC table) resource and a service-specific port map.
- Mapping to service-specific arrival and departure frame modification instructions.
- Mapping to a protection group resource, enabling fast failover of multiple services using a single group-level protection action. When using this protection mechanism, each service may retain individual frame modification instructions for both working and protect states.
- Service-specific QoS handling instructions including remarking and hierarchical QoS treatment.
- OAM and control protocol handling, including mapping to Versatile OAM Engine (VOE) hardware resources.
- Per-service arrival, departure, and discard statistics. Multiple services can map to one statistics set. Up to eight statistics sets can be assigned to a service to implement the per-CoS per-EVC MEF bandwidth profile.

3.2.4 Security and Control Protocol Classification

Before being passed on to the Layer 2 forwarding, all frames are inspected by the VCAP IS2 for security enforcement and control protocol actions.

The action associated with each IS2 entry (programmed into the IS2 action RAM) includes frame filtering, single leaky bucket rate limiting (frame or byte based), snooping to CPU, redirecting to CPU, mirroring, time stamping, and accounting. Although the VCAP IS2 is located in the ingress path of the device, it has both ingress and egress capabilities.

The VCAP IS2 engine embeds powerful protocol awareness for well-known protocols such as LLC, SNAP, ARP, IPv4, IPv6, and UDP/TCP. IPv6 is supported with full matching against both the source and the destination IP addresses.

For each frame, up to two lookup keys are generated and matched against the VCAP entries.

VCAP CLM can enable OAM processing, either by copy or redirect to the CPU or by assigning the frame to a VOE for hardware OAM processing. VCAP IS2 can enable OAM processing by copy or redirect to the CPU.

3.2.5 Policing

Each frame is subject to one or more of the following policing operations.

- Dual-rate service policers. Service classification selects which service policer to use.
- Dual-rate bundle policers. Service classification selects which bundle policer to use. Bundle policers are placed immediately after the service policers and can police one or more services.
- Single-rate priority policers. Ingress port number and QoS class determine which policer to use.
- Single-rate port policers. Ingress port number determines which policer to use.
- VCAP single-rate policers. IS2 action selects which VCAP single-rate policer to use.
- VCAP dual-rate policers. IS2 action selects which VCAP dual-rate policer to use.
- Single-rate global storm policers. Frame characteristicS such as broadcast, unicast, multicast, or learn frame select which policer to use.
- Single-rate service broadcast, unicast, and multicast (BUM) policers for flooded frames. Service classification and destination MAC address select which policer to use.

Policers can measure frame rates (frames per second) or bit rates (bits per second). Service frames (frames classified to a service) can trigger the following policers.

- One BUM policer
- One service policer
- One bundle policer
- One VCAP policer

- Eight storm policers

Non-service frames can trigger the following policers:

- One BUM policer
- One priority policer or one dual-rate VCAP policer
- One port policer
- One VCAP policer
- Two port policers
- Eight storm policers

Dual-rate policers are MEF-compliant, supporting both color-blind and color-aware operation. The initial frame color is derived from the drop precedence level from the frame classification. The MEF coupling mode is also configurable for each policer.

Dual-rate policers ensure Service Level Agreement (SLA) compliance. The outcome of this policing operation is to mark each accepted frame as in-profile (green) or out-of-profile (yellow). Yellow frames are treated as excess or discard-eligible and green frames are committed. Frames that exceed the yellow/excess limits are discarded (red).

Each frame is counted in associated statistics counters reflecting the service and the frame's color (green, yellow, red). The statistics counters count bytes and frames.

The two port policers and eight storm policers can be programmed to limit different types of traffic, such as CPU-forwarded frames, learn frames, known or unknown unicast, multicast, or broadcast.

3.2.6 Layer 2 Forwarding

After the policers, the Layer 2 forwarding block of the analyzer handles all fundamental Layer 2 forwarding operations and maintains the associated MAC table, the VLAN table, the ISDX table, and the aggregation table. The device implements an 8K MAC table and a 4K VLAN table.

The main task of the analyzer is to determine the destination port set of each frame. The Layer 2 forwarding decision is based on various information such as the frame's ingress port, source MAC address, destination MAC address, the VLAN identifier, and the ISDX, as well as the frame's VCAP actions, mirroring, and the destination port's link aggregation configuration.

Layer 2 forwarding of unicast and Layer 2 multicast frames is based on the destination MAC address and the VLAN.

The switch can also L2-forward IPv4 and IPv6 multicast frames using additional Layer-3 information, such as the source IP address. The latter enables source-specific IPv4 multicast forwarding (IGMPv3) and source-specific IPv6 multicast forwarding (MLDv2). This process of L2-forwarding multicast frames using Layer 3 information is called "snooping", which is different from L3-forwarding (routing) of multicast frames.

The following describes some of the contributions to the Layer 2 forwarding.

- **VLAN Classification** VLAN-based forward filtering includes source port filtering, destination port filtering, VLAN mirroring, and asymmetric VLANs.
- **Service Classification** Service-based forward filtering includes destination port filtering and forced forwarding.
- **Security Enforcement** The security decision made by the VCAP IS2 can, for example, redirect the frame to the CPU based on security filters.
- **MSTP** The VLAN identifier maps to a Multiple Spanning Tree instance, which determines MSTP-based destination port filtering.
- **MPLS** Destination set determined by processing MPLS labels.
- **MAC Addresses** Destination and source MAC address lookups in the MAC table determine if a frame is a learn frame, a flood frame, a multicast frame, or a unicast frame.
- **Learning** By default, the device performs hardware learning on all ports. However, certain ports could be configured with secure learning enabled, where an incoming frame with unknown source MAC address is classified as a "learn frame" and is redirected to the CPU. The CPU performs the learning decision and also decides whether the frame is forwarded. Learning can also be disabled, such as for E-Line services. If learning is disabled, it does not matter if the source MAC address is in the MAC table.

- **Link Aggregation** A frame targeted to a link aggregate is processed to determine which physical port of the link aggregate group the frame must be forwarded to.
- **Mirroring** Mirror probes may be set up in different places in the forwarding path for monitoring purposes. As part mirroring, a copy of the frame is sent either to the CPU or to another port.

3.2.7 Layer 3 Forwarding

The device supports Layer 3 forwarding (routing), which is performed by the analyzer. With Layer 3 forwarding, the IPv4 or IPv6 addresses determine the destination port set and Layer 3 processing is performed on IP header. The Layer 3 forwarding process also replaces the arrival Layer-2 Ethernet header (including MAC addresses and VLAN tags) with a new Layer 2 Ethernet header after departure from the switch.

The analyzer supports 32 router legs, and supports Virtual Router Redundancy Protocol (VRRP). Ingress router legs are addressed using classified arrival VLAN and DMAC address.

If an arrival frame is determined to belong to an ingress router leg and routing is enabled, Layer 3 forwarding is applied. Note that this is in addition to any Layer 2 forwarding, so for example, an arrival multicast frame may be Layer 2 forwarded on the classified arrival VLAN and also Layer 3 forwarded to one or more additional VLANs. IP unicast frames only use Layer 2 forwarding or Layer 3 forwarding, but never both. Layer 3 forwarding always uses the classified arrival VLAN and does not use the ISDX.

Layer 3 forwarding first checks the IP header for validity. IP header checks include IP version, header length, and header checksum. The Time-to-Live (TTL) or Hop Limit values are also checked for validity and decremented if the packet is forwarded.

The analyzer then searches the Longest Prefix Match (LPM) table for an exact match of the destination IP address. If there is not an exact match, the table is searched for the best partial match. If there is no partial match in the LPM table, the frame is Layer 3 forwarded to the location of the default gateway.

With Layer 3 forwarding, each egress frame copy uses an egress router leg to determine the per-copy egress encapsulation. There can be up to 16 egress router legs associated with one forwarding entry in support of Equal Cost Multipath (ECMP) functionality, where multiple forwarding paths are used between adjacent routers for increased forwarding bandwidth.

Reverse path forwarding (RPF) is optionally performed on multicast and unicast (uRPF) packets as a security check. This source IP address check helps detect and prevent address spoofing.

Multicast frames can be Layer 2 forwarded on the arrival VLAN as well as Layer 3 forwarded to different VLANs. Layer 3 forwarding of IP multicast is different from Layer 2 forwarding of IP multicast using IGMP/MLD snooping in the following ways:

- IP header checks and TTL processing are performed
- The Ethernet header is swapped
- The egress VLANs may be different from the ingress VLANs
- Multiple frame copies can be generated per physical port

Network control such as ICMP and ARP are redirected to the CPU, along with malformed packets, packets with expired TTL, and packets with options.

3.2.8 Shared Queue System and Hierarchical Scheduler

The analyzer provides the destination port set of a frame to the shared queue system. It is the queue system's task to control the frame forwarding to all destination ports.

The shared queue system embeds memory that can be shared between all queues and ports. Frames are mapped to queues through a programmable mapping function allowing ingress ports, egress ports, QoS classes, and services to be taken into account. The sharing of resources between queues and ports is controlled by an extensive set of thresholds.

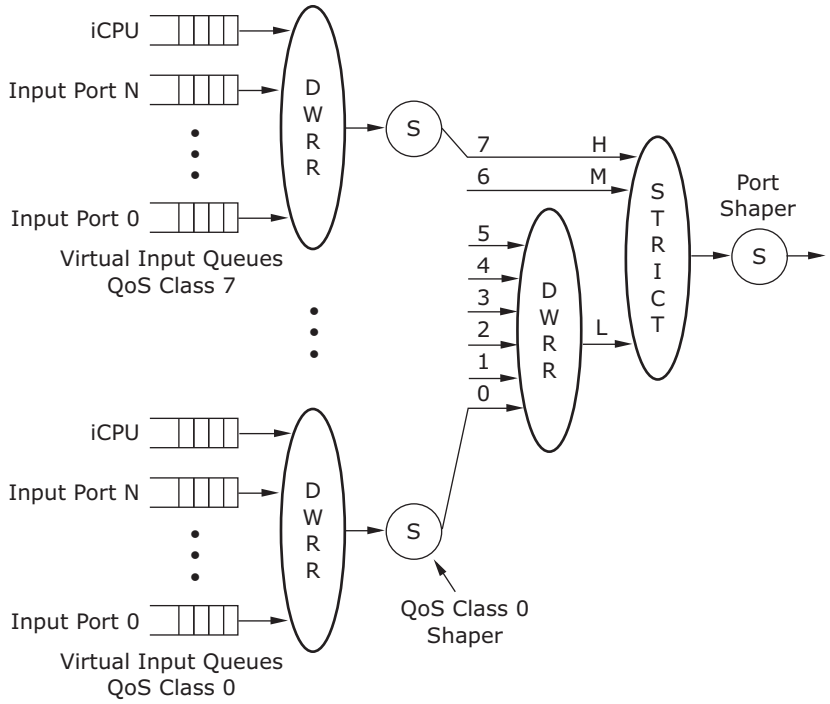
Each egress port implements default schedulers and shapers as shown in the following illustration. Per egress port, the scheduler sees the outcome of aggregating the egress queues (one per ingress port per QoS class) into eight QoS classes.

3.2.8.1 Class-Based Queuing (Default Configuration)

By default, aggregation is done in a Deficit Weighted Round Robin fashion (DWRR) with equal weights to all ingress ports within a QoS class, so byte-accurate weighted round robin scheduling between ingress ports is performed for each QoS class.

The following illustration shows the default scheduler-shaper configuration. Hierarchical scheduling-shaping is also possible. All shapers shown are dual leaky bucket shapers.

Figure 11 • **Default Scheduler-Shaper Configuration**



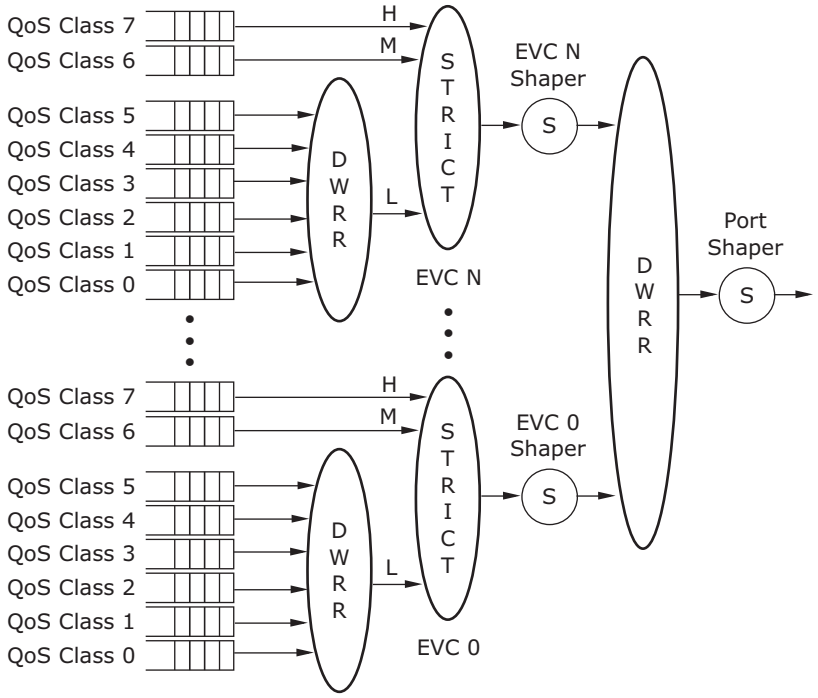
Scheduling between QoS classes within the port can use one of the three following methods:

- Strict Priority scheduling. Frames with the highest QoS class are always transmitted before frames with lower QoS class.
- Deficit Weighted Round Robin (DWRR) scheduling. Each QoS class serviced using DWRR sets a DWRR weight ranging from 0 to 31.
- Combination of Strict Priority and DWRR scheduling. The default configuration is shown, where QoS classes 7 and 6 use strict priority while QoS classes 5 through 0 use DWRR. But any split between strict and DWRR QoS classes can be configured.

3.2.8.1.1 Hierarchical QoS (H-QoS)

Hierarchical Quality of Service (H-QoS) egress scheduling can be configured as shown in the following illustration. Scheduling QoS within each EVC is performed as previously described for scheduling QoS within a port. Each EVC can optionally be shaped using a MEF-compliant dual-rate shaper. EVCs are then scheduled at the port level using a DWRR scheduler. The physical port can optionally be shaped using a MEF-compliant dual-rate shaper.

Figure 12 • **Hierarchical Scheduler-Shaper Configuration**



3.2.9 Rewriter and Frame Departure

Before transmitting the frame on the egress line, the rewriter can modify selected fields in the frame such as Ethernet headers and VLAN tags, MPLS labels, IPv4/IPv6 fields, OAM fields, PTP fields, and FCS. The rewriter also updates the frame's COS and color indications such as DEI, PCP, MPLS TC, and DSCP. Departure statistics are also kept based on either the ESDX from the VCAP ES0 lookup or the classified VLAN.

- **Basic VLAN Tagging Operations (Port-based)** By default, the egress VLAN actions are determined by the egress port settings and classified VLAN. These include basic VLAN operations such as pushing a VLAN tag, untagging for specific VLANs, and simple translations of DEI, PCP, and DSCP.
- **Advanced VLAN Tagging Operations (Port-based and Service-based)** By using the egress VCAP ES0, significantly more advanced VLAN tagging operations can be achieved. ES0 enables pushing up to three VLAN tags and flexible VLAN tag translation for per-EVC VLAN tag/TPID, PCP, DEI, and DSCP control. The lookup by VCAP ES0 includes classified VLAN (which is VSI for services), ISDX, egress port, and COS/color indications.
- **MPLS Operations** MPLS push, pop, and swap are always handled by VCAP ES0. Up to three MPLS labels and one control word can be popped and pushed, as well as an Ethernet link layer header. Per-EVC/LSP/PW, control is supported for Ethernet link layer encapsulation and VID, MPLS labels, and TC bit remarking.
- **Layer 3 Forwarding (Routing) Operations** Supports per-router-leg control of Ethernet link layer encapsulation and VID, as well as modification of other Layer 3 fields such as IPv4 TTL, IPv4 checksum, and IPv6 hop limit.
- **OAM Operations** OAM Protocol data unit (PDU) modifications include insertion of time stamps and frame counters, as well as OAM Message/Reply swap operations. General SMAC/DMAC swapping is also supported for OAM and non-OAM frames.
- **PTP (IEEE 1588) Operations** PTP PDU modifications include insertion/update of time stamps and correction fields, as well as updating the UDP checksum.

The rewriter pads frames to minimum legal size if needed, and updates the FCS if the frame was modified before the frame is transmitted.

The egress port module controls the flow control exchange of pause frames with a neighboring device when the interconnection link operates in full-duplex flow control mode.

3.2.10 CPU Port Module

The CPU port module (DEVCPU) contains eight CPU extraction queues and two CPU injection queues. These queues provide an interface for exchanging frames between the internal CPU system and the switch core. An external CPU using the serial interface can also inject and extract frames to and from the switch core by using the CPU port module.

In addition, any Ethernet interface on the device can be used for extracting and injecting frames. The Ethernet interface used in this way is called the Node Processor Interface (NPI) and is typically connected to an external CPU.

Injected frames may be prepended with an injection header to control processing and forwarding of these frames. Extracted frames may be prepended with an extraction header to supply frame arrival and classification information associated with each frame. These headers may be used by internal CPU or external CPU.

The switch core can intercept a variety of different frame types and copy or redirect these to the CPU extraction queues. The classifier can identify a set of well-known frames, such as IEEE reserved destination MAC addresses (BPDUs, GARPs, CCM/Link trace), as well as IP-specific frames (IGMP, MLD). Security VCAP IS2 provides another flexible way of intercepting all kinds of frames, such as specific OAM frames, ARP frames or explicit applications based on TCP/UDP port numbers. In addition, frames can be intercepted based on the MAC table, the VLAN table, or the learning process.

Whenever a frame is copied or redirected to the CPU, a CPU extraction queue number is associated with the frame and used by the CPU port module when enqueuing the frame into the eight CPU extraction queues. The CPU extraction queue number is programmable for every interception option in the switch core.

3.2.11 Synchronous Ethernet and Precision Time Protocol (PTP)

The device supports Synchronous Ethernet and IEEE 1588 Precision Time Protocol (PTP) for synchronizing network timing throughout a network.

Synchronous Ethernet allows for the transfer of precise network timing (frequency) using physical Ethernet link timing. Each switch port can recover its ingress clock and output the recovered clock to one of up to four recovered clock output pins. External circuitry can then generate a stable reference clock input used for egress and core logic timing.

The Precision Time Protocol (PTP) allows for the transfer of precise network timing (frequency) and time of day (phase) using Ethernet packets. PTP can operate as a one-step clock or a two-step clock. For one-step clocks, a precise time is calculated and stamped directly into the PTP frames at departure. For two-step clocks, a precise time is simply recorded and provided to the CPU for further processing. The CPU can then initiate a follow-up message with the recorded timing.

The device supports PTP Delay_req/Delay_resp processing in hardware where the Delay_req frame is terminated, and a Delay_resp frame is generated based on the request. In addition to updating relevant PTP time stamps and message fields, the frame encapsulation can be changed. This includes swapping and rewriting MAC addresses, IP addresses, and TCP/UDP ports, and updating the IPv4 TTL or IPv6 hop limit.

The device also supports generation of PTP Sync frames using the automatic frame injector (AFI) functionality.

PTP is supported for a range of encapsulations including the following:

- PTP over Ethernet
- PTP over UDP over IPv4/IPv6 over Ethernet
- Either of the above encapsulations over MPLS pseudowire over Ethernet
- PTP over UDP over IPv4/IPv6 over MPLS over Ethernet

Two separate timing domains are supported: one for Synchronous Ethernet and data path forwarding, and one for PTP timing synchronization. This gives the system designer control over how these two timing architectures interact.

3.2.12 Ethernet and MPLS OAM

Hardware MEP functions are implemented for Ethernet and MPLS OAM using Versatile OAM engines (VOE). Hierarchical MEP functionality is supported up to four layers deep, including a mix of Up and Down MEPs. Each incoming OAM frame is processed by the assigned VOE, which includes verification of the frame contents against configurations, forwarding based on the MEG level, and processing according to the received MPLS or Ethernet OAM type.

In combination with the MEPs, hardware MIPs can implement a mix of Down-MIP and Up-MIP half functions, which include verifying the OAM frame contents against configurations and providing the frame to the CPU for further processing.

3.2.12.1 Ethernet OAM

The switch supports various Ethernet OAM layer interactions as defined in ITU-T Y.1731, IEEE 802.3, and IEEE 802.1ag. The following Ethernet OAM functions are supported in hardware. Other Ethernet OAM functions are supported, but require CPU interaction.

- Continuity check CCM frame generation and checking with sub-millisecond transmission periods, including support for RDI and sequence number functions.
- Loopback with frame generation and checking for both LBM and LBR, and the Loopback Responder function receiving LBM and replying with LBR at full rate including MAC address swapping.
- Frame loss measurements with hardware-accurate counter readings supporting single-ended and dual-ended measurements. Single-ended loss measurements use LMM and LMR, whereas dual-ended loss measurements use CCM.
- Frame delay measurements with hardware-accurate time stamps supporting one-way and two-way delay measurements. One-way delay measurements use 1DM, whereas two-way delay measurements DMM and DMR.
- Test frame generation and checking.

3.2.12.2 MPLS OAM

The switch supports the following MPLS and pseudowire OAM. Hardware supports MPLS OAM channel detection for all formats listed. Hardware also provides BFD OAM generation and checking with sub-millisecond transmission periods.

Other MPLS OAM functions are supported, but require CPU interaction.

- Ethernet OAM: As described previously, inside MPLS pseudowire.
- Pseudowire OAM: The OAM channel for VCCV types 1-3 detected per IETF RFC5085, also “type 4” OAM channel detected using MPLS-TP GAL.
- MPLS-TP OAM: GAL/G-ACH detected as per IETF RFC5586, which detects the OAM channel as described in both ITU-T G.8113.2 (MPLS-TP using tools defined for MPLS, based on IETF MPLS OAM) and ITU-T G.8113.1 (MPLS-TP in Packet Transport Network, based on ITU-T Y.1731 OAM)
- MPLS-TP Bidirectional Forwarding Detection (BFD): ACH-encapsulated within pseudowires and LSPs as per IETF RFC6428.

3.2.13 CPU Subsystem

The device contains a powerful, 500 MHz MIPS24KEc-compatible microprocessor, a high bandwidth Ethernet Frame DMA engine, and a DDR3/DDR3L controller supporting up to 1 gigabyte (GB) of memory. This complete system-on-chip supports Linux or embedded operating systems, enabling full management of the switch and advanced software applications.

The device supports external CPU register access by the on-chip PCIe 1.x endpoint controller, by specially formatted Ethernet frames on the NPI port (Versatile Register Access Protocol), or by register access interface using SPI protocol. External CPUs can inject or extract Ethernet frames by the NPI port, by PCIe DMA access, or by register read/writes (using any register-access interface).

3.3 Frame Headers

This section describes the internal header formats used within the device that are visible in frames to and from the CPU, NPI port, and in frames exchanged between devices in multichip configurations.

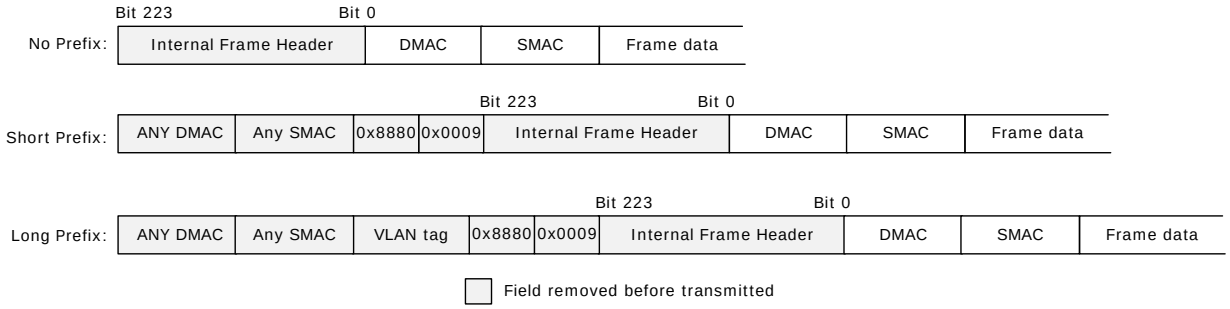
The header formats are internal frame header (IFH) and VStaX header.

- **Internal frame header (IFH)** IFH is used when extracting frames to CPU and injecting frames from CPU. The IFH can also be inserted into frames transmitted on the NPI port. The IFH includes a VStaX header.
- **VStaX header** The VStaX header can be used for transmission to and from an NPI port.

3.3.1 Internal Frame Header Placement

The following illustration shows internal frame header placement.

Figure 13 • **Frame with Internal Frame Header**



Frames are injected without prefix from internal CPU or directly attached CPU.

Frames can be injected with a prefix to accommodate CPU injection from a front port (that may be directly attached), controlled by ASM:CFG:PORT_CFG.INJ_FORMAT_CFG.

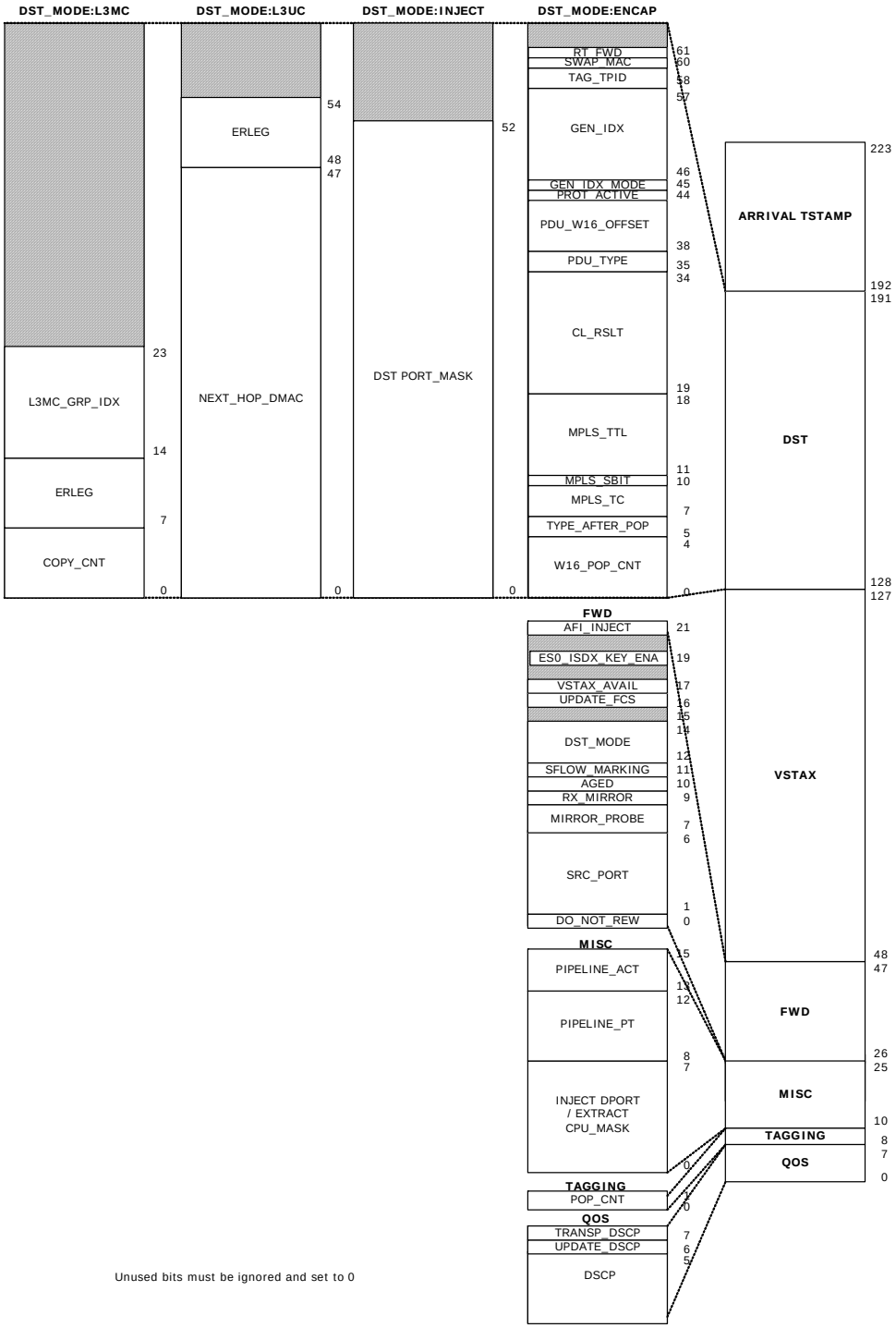
The internal frame header and optional prefix is removed before transmitting the frame on a port. It is only visible to the user when injecting and extracting frames to or from the switch core and optionally when sending/receiving frames using the NPI port.

It is possible to configure a port to transmit frames with internal frame headers using REW:COMMON:PORT_CTRL.KEEP_IFH_SEL. It is also possible to only transmit CPU redirected frames with Internal frame headers using REW:COMMON:GCPU_CFG.GCPU_KEEP_IFH.

3.3.2 Internal Frame Header Layout

The internal frame header is 28 bytes long. The following illustration shows the layout.

Figure 14 • **Internal Frame Header**



The following table shows the internal frame header fields, including information whether a field is relevant for injection, extraction, or both.

Table 2 • **Internal Frame Header Fields**

Field Category	Field Name	Bit	Width	Description
TS	TSTAMP	223:192	32 bits	Arrival time stamp in nanoseconds.

Table 2 • **Internal Frame Header Fields *(continued)***

Field Category	Field Name	Bit	Width	Description
DST when FWD.DST_MODE is INJECT	RSV	191:181	11 bits	Reserved field. Must be set to 0.
	DST PORT_MASK	180:128	53 bits	Injection destination port mask where each bit representing a physical port (only used for injection).
DST when FWD.DST_MODE is ENCAP	RSV	191:190	2 bits	Reserved field. Must be set to 0.
	RT_FWD	189	1 bit	Update IP4 TTL and chksum/ip6 Hopcnt.
	SWAP_MAC	188	1 bit	Instruct rewriter to swap MAC addresses.
	TAG_TPID	187:186	2 bits	Tag protocol IDs. 0: Standard TPID as specified in VSTAX.TAG.TAG_TYPE. 1: custom1 stag TPID. 2: custom2 stag TPID. 3: custom3 stag TPID.
	RSV	185:184	2 bits	Reserved field. Must be set to 0.
	GEN_IDX	183:174	10 bit	Generic index. VSI when GEN_IDX_MODE = 1.
	GEN_IDX_MODE	173	1 bit	0: Reserved. 1: VSI.
	PROT_ACTIVE	172	1 bit	Protect is active.
	PDU_W16_OFFSET	171:166	6 bits	PDU WORD16 (= 2 bytes) offset from W16_POP_CNT to Protocol data unit (PDU).
	PDU_TYPE	165:163	3 bits	PDU type used to handle OAM, PTP and SAT. 0: None. 1: OAM_Y1731. 2: OAM_MPLS_TP. 3: PTP. 4: IP4_UDP_PTP. 5: IP6_UDP_PTP. 6: Reserved. 7: SAM_SEQ.
	CL_RSLT	162:147	16 bits	Classified MATCH_ID combined from VCAP CLM and VCAP IS2. Used if the frame is forwarded to the CPU.
	MPLS_TTL	146:139	8 bits	TTL value for possible use in MPLS label.
	MPLS_SBIT	138	1 bit	SBIT of last popped MPLS label. for possible use in MPLS label.
	MPLS_TC	137:135	3 bits	TC value for possible use in MPLS label.

Table 2 • **Internal Frame Header Fields *(continued)***

Field Category	Field Name	Bit	Width	Description
DST when FWD.DST_MODE is ENCAP	TYPE_AFTER_POP	134:133	2 bits	0: ETH - Normal Ethernet header, starting with DMAC. 1: CW. First 4 bits: 4: IPv4 header. 6: IPv6 header. Others: MPLS CW (see RFC 4385) 2: MPLS. MPLS shim header follows.
	W16_POP_CNT	132:128	5 bits	Number of WORD16 (= 2 bytes) to be popped by rewriter, starting from beginning of frame.
DST when FWD.DST_MODE is “L3UC”	RSV	191:183	9 bits	Reserved field. Must be set to 0.
	ERLEG	182:176	7 bits	Egress router leg.
	NEXT_HOP_DMAC	175:128	48 bits	Next hop DMAC. Only used for unicast routing.
DST when FWD.DST_MODE is “L3MC”	RSV	191:152	40 bits	Reserved field. Must be set to 0.
	L3MC_GRP_IDX	151:142	10 bits	IP multicast group used for L3 multicast copies.
	ERLEG	141:135	7 bits	Egress router leg.
	COPY_CNT	134:128	7 bits	Number of multicast routed copies. Only used for multicast routing.
VSTAX		127:48	80 bits	VStaX. See VStaX Header, page 29.

Table 2 • **Internal Frame Header Fields *(continued)***

Field Category	Field Name	Bit	Width	Description
FWD	AFI_INJ	47	1 bit	Injected into AFI.
	RSV	46	1 bit	Reserved field. Must be set to 0.
	ES0_ISDX_KEY_ENA	45	1 bit	Controls use of ISDX in ES0 key.
	RSV	44	1 bit	Reserved field. Must be set to 0.
	VSTAX_AVAIL	43	1 bit	Received by ANA with valid VSTAX section. Extract: Frame received by ANA with valid VSTAX section. Inject: Frame to be forwarded based on VSTAX section.
	UPDATE_FCS	42	1 bit	Forces update of FCS. 0: Does not update FCS. FCS is only updated if frame is modified by hardware. 1: Forces unconditional update of FCS.
	RSV	41	1 bit	Reserved field. Must be set to 0.
	DST_MODE	40:38	3 bits	Controls format of IFH.DST. 0: ENCAP 1: L3UC routing 2: L3MC routing 3: INJECT Others: Reserved
	SFLOW_MARKING	37	1 bit	Frame forwarded to CPU due to sFlow sampling.
	AGED	36	1 bit	Must be set to 0. Set if frame is aged by QSYS. Only valid for extraction.
	RX_MIRROR	35	1 bit	Signals that the frame is Rx mirrored.
	MIRROR_PROBE	34:33	2 bits	Signals mirror probe for mirrored traffic. 0: Not mirrored. 1-3: Mirror probe 0-2.
	SRC_PORT	32:27	6 bits	Physical source port number. May be set by CPU to non-CPU port number to masquerade another port.
	DO_NOT_REW	26	1 bit	Controlled by CPU or ANA_CL:PORT:QOS_CFG.KEEP_ENA and prevents the rewriter from making any changes of frames sent to front ports when set.

Table 2 • **Internal Frame Header Fields *(continued)***

Field Category	Field Name	Bit	Width	Description
MISC	PIPELINE_ACT	25:23	3 bits	Pipeline action specifies if a frame is injected, extracted, or discarded. 0: None 1: INJ 2: INJ_MASQ 3: Reserved 4: XTR 5: XTR_UPMEP 6: LBK_ASM 7: LBK_QS
	PIPELINE_PT	22:18	5 bits	Pipeline point specifies the location where a frame is injected, extracted, or discarded. 0: None 1: ANA_VRAP 2: ANA_PORT_VOE 3: ANA_CL 4: ANA_CLM 5: ANA_IPT_PROT 6: ANA_OU_MIP 7: ANA_OU_SW 8: ANA_OU_PROT 9: ANA_OU_VOE 10: ANA_MID_PROT 11: ANA_IN_VOE 12: ANA_IN_PROT 13: ANA_IN_SW 14: ANA_IN_MIP 15: ANA_VLAN 16: ANA_DONE 17: REW_IN_MIP 18: REW_IN_SW 19: RE_IN_VOE 20: REW_OU_VOE 21: REW_OU_SW 22: REW_OU_MIP 23: REW_SAT 24: REW_PORT_VOE 25: REW_VRAP
	CPU_MASK	17:10	8 bits	CPU extraction queue mask.
TAGGING	POP_CNT	9:8	2 bits	Controlled by CPU or ANA_CL:PORT:VLAN_CTRL.VLAN_POP_CNT or CLM VLAN_POP_CNT_ENA and causes the rewriter to pop the signaled number of consecutive VLAN tags. If ENCAP.W16_POP_CNT >0 and TYPE_AFTER_POP = ETH POP_CNT applies to inner Ethernet layer.

Table 2 • **Internal Frame Header Fields *(continued)***

Field Category	Field Name	Bit	Width	Description
QOS	TRANSP_DSCP	7	1 bit	Controlled by CPU or ANA_CL:PORT:QOS_CFG. DSCP_KEEP_ENA and prevents the rewriter from remapping DSCP values of frames sent to front ports when set.
	UPDATE_DSCP	6	1 bit	Controlled by CPU, ANA_CL:PORT:QOS_CFG DSCP_REWR_MODE_SEL, CLM DSCP_ENA or ANA_CL:MAP_TBL: SET_CTRL.DSCP_ENA and causes the rewriter to update the frame's DSCP value with IFH.QOS.DSCP for frames sent to front ports when set.
	DSCP	5:0	6 bits	DSCP value.

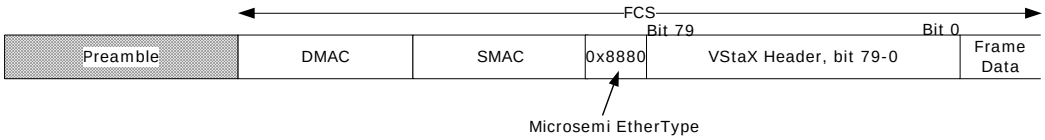
The IFH is only used to control the rewriter on front ports or send to CPU. It is not transmitted with the frame. For CPU ports, the information in the extraction IFH is for reference purposes only.

3.3.3 VStaX Header

When frames are sent to or from the NPI port, or when connecting multiple switches together, an optional VStaX header is inserted into the frames.

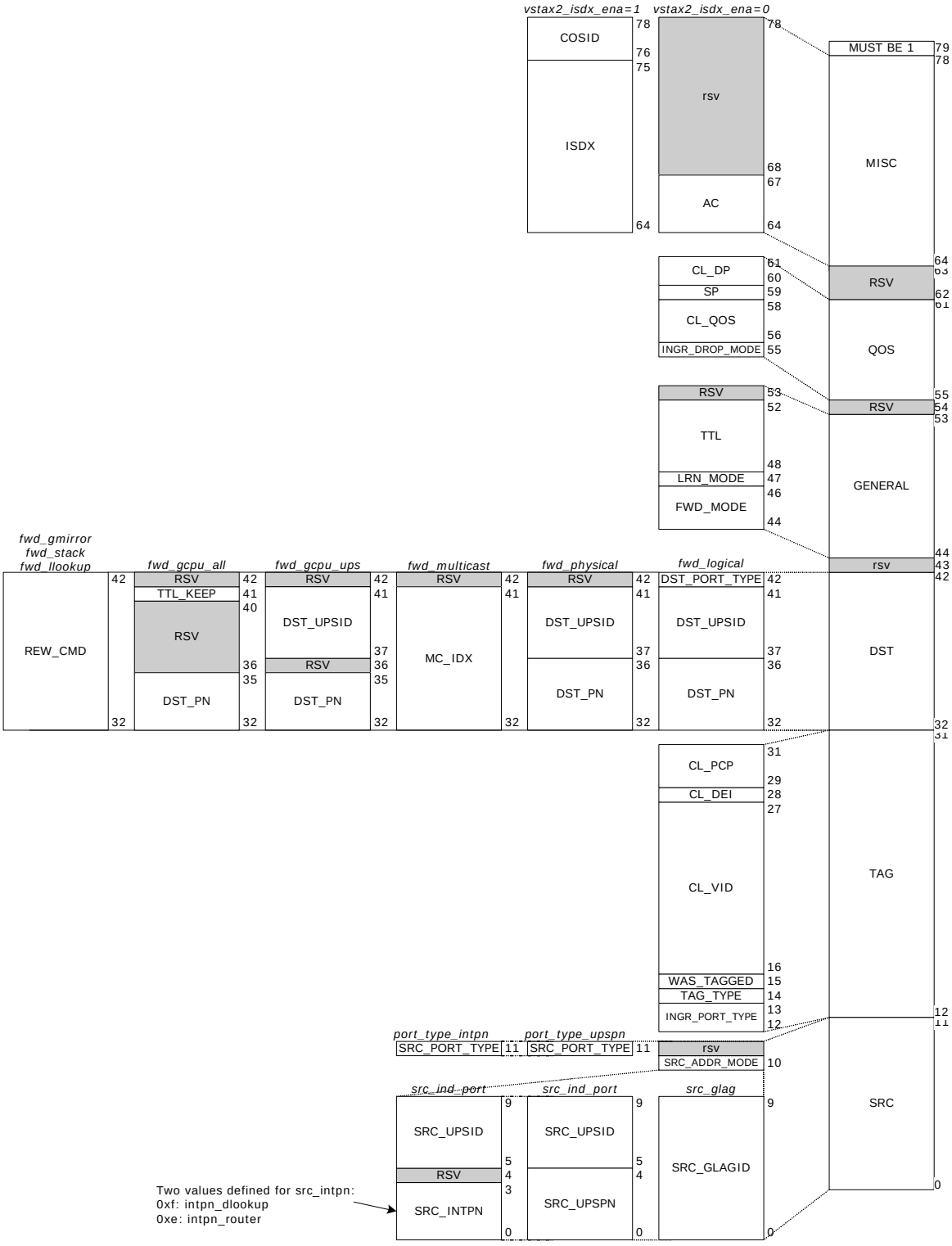
The VStaX header consists of a 10-byte payload and is preceded by 2 bytes for the Microsemi EtherType (0x8880). That is, a total of 12 bytes are inserted into the frame, as shown in the following illustration.

Figure 15 • **Frame With VStaX Header**



The layout of the 10-byte VStaX header is shown in the following illustration. In VStaX context, each switch in a stack is termed a unit.

Figure 16 • **VStaX Header Layout**



The following table describes each field in the VStaX header.

Table 3 • **VStaX Header Fields**

Field Category	Field Name	Bit	Width	Description
Reserved	RSV	79	1 bit	Reserved field. Must be set to 1 when injecting frames and must be checked on extraction.
MISC when ANA_AC:PS_COMMON.VSTAX2_MISC_ISDX_ENA=1	COSID	78:76	3 bits	Class of service.
	ISDX	75:64	12 bits	Ingress service index.
MISC when ANA_AC:PS_COMMON.VSTAX2_MISC_ISDX_ENA=0	RSV	78:68	11 bits	Reserved field. Must be set to 0 when injecting frames from CPU and ignored on extraction.
	AC	67:64	4 bits	GLAG aggregation code
Reserved	RSV	63:62	2 bits	Reserved field. Must be set to 0.
QOS	CL_DP	60:61	2 bits	Classified drop precedence level.
	SP	59	1 bit	Super priority. Identifies frames to be forwarded between CPUs of neighboring units, using egress and ingress super priority queues in the assembler and disassembler.
	CL_QOS (IPRIO)	58:56	3 bits	Classified quality of service value (internal priority).
	INGR_DROP_MODE	55	1 bit	Congestion management information. 0: Ingress front port is in flow control mode. 1: Ingress front port is in drop mode.
Reserved	RSV	54	1 bit	Reserved field. Must be set to 0.
General	RSV	53	1 bit	Reserved field. Must be set to 0.
	TTL	52:48	5 bits	Time to live.
	LRN_MODE	47	1 bit	0: Normal learning. SMAC and VID of the frame is subject to learning. 1: Skip learning. Do not learn SMAC and VID of the frame.
	FWD_MODE	46:44	3 bits	Forward mode. Encoding: 0x0: FWD_LLOOKUP: Forward using local lookup in every unit. 0x1: FWD_LOGICAL: Forward to logical front port at specific UPS. 0x2: FWD_PHYSICAL: Forward to physical front port at specific UPS. 0x3: FWD_MULTICAST: Forward to ports part of multicast group. 0x4: FWD_GMIRROR: Forward to global mirror port. 0x5: FWD_GCPU_UPS: Forward to GCPU of specific UPS. 0x6: FWD_GCPU_ALL: Forward to all GCPUs. 0x7: Reserved.
Reserved	RSV	43	1 bit	Reserved field. Must be set to 0.

Table 3 • **VStaX Header Fields *(continued)***

Field Category	Field Name	Bit	Width	Description
VSTAX.DST when VSTAX.FWD_MODE is FWD_LLOOKUP or FWD_GMIRROR	REW_CMD	42:32	11 bits	VCAP IS2 action REW_CMD.
VSTAX.DST when VSTAX.FWD_MODE is FWD_LOGICAL	DST_PORT_TYPE	42	1 bit	Destination port type. Encoding: 0: Front port 1: Internal port
	DST_UPSID	41:37	5 bits	Destination unit port set ID.
	DST_PN	36:32	5 bits	Logical destination port at unit identified by dst_upsid.
VSTAX.DST when VSTAX.FWD_MODE is FWD_PHYSICAL	RSV	42	1 bit	Reserved field. Must be set to 0.
	DST_UPSID	41:37	5 bits	Destination unit port set ID.
	DST_PN	36:32	5 bits	Physical destination port at unit identified by dst_upsid.
VSTAX.DST when VSTAX.FWD_MODE is FWD_MULTICAST	RSV	42	1 bit	Reserved field. Must be set to 0.
	MC_IDX	41:32	10 bits	Forward to ports part of this multicast group index.
VSTAX.DST when VSTAX.FWD_MODE is FWD_GCPU_UPS	RSV	42	1 bit	Reserved field. Must be set to 0.
	DST_UPSID	41:37	5 bits	Destination unit port set ID.
	RSV	36	1 bit	Reserved field. Must be set to 0.
	DST_PN	35:32	4 bits	CPU destination port at unit identified by dst_upsid.
VSTAX.DST when VSTAX.FWD_MODE is FWD_GCPU_ALL	RSV	42	1 bit	Reserved field. Must be set to 0.
	TTL_KEEP	41	1 bit	Special TTL handling used for neighbor discovery.
	RSV	40:36	5 bits	Reserved field. Must be set to 0.
	DST_PN	35:32	4 bits	CPU destination port at unit identified by dst_upsid.
TAG	CL_PCP	31:29	3 bits	Classified priority code point value.
	CL_DEI	28	1 bit	Classified drop eligible indicator value.
	CL_VID	27:16	12 bits	Classified VID.
	WAS_TAGGED	15	1 bit	If set, frame was VLAN-tagged at reception.
	TAG_TYPE	14	1 bit	Tag type. 0: C-tag (EtherType 0x8100). 1: S-tag (EtherType 0x88A8).
	INGR_PORT_TYPE	13:12	2 bits	Ingress ports type for private VLANs/Asymmetric VLANs. 00: Promiscuous port. 01: Community port. 10: Isolated port.

Table 3 • **VStaX Header Fields *(continued)***

Field Category	Field Name	Bit	Width	Description
SRC	SRC_ADDR_MODE	10	1 bit	0: src_ind_port: Individual port. Source consists of src_upsid and src_upspn. 1: src_glag: Source consists of src_glag.
	SRC_PORT_TYPE	11	1 bit	Only applicable if src_addr_mode==src_ind_port. Reserved and must be set to 0 if src_addr_mode==src_glag. 0: port_type_upspn. 1: port_type_intpn.
	SRC_UPSID	9:5	5 bits	If src_addr_mode = src_ind_port: ID of stack unit port set, where the frame was initially received.
	SRC_UPSPN	4:0	5 bits	If src_addr_mode = src_ind_port and src_port_type = port_type_upspn: Logical port number of the port (on source ups), where the frame was initially received.
	SRC_INTPN	3:0	4 bits	If src_addr_mode = src_ind_port and src_port_type = port_type_intpn: Internal port number.
	SRC_GLAGID	0	5 bits	If src_addr_mode = src_glag: ID of the GLAG.

3.4 Port Numbering and Mappings

The switch core contains an internal port numbering domain where ports are numbered from 0 through 14. A port connects to a port module, which connects to an interface macro, which connects to I/O pins on the device. The port modules are DEV2G5. The interface macros are SERDES1G, SERDES6G, or a copper transceiver.

Some ports are internal to the device and do not connect to port modules or interface macros. For example, internal ports are used for frame injection and extraction to the CPU queues.

The following table shows the default port numbering and how the ports map to port modules and interface macros.

Table 4 • **Default Port Numbering and Port Mappings**

Port Number	Port Module	Maximum Port Speed	Interface Macro
0	DEV2G5_0	1 Gbps	SERDES1G_0 or CUPHY_0
1	DEV2G5_1	1 Gbps	SERDES1G_1 or CUPHY_1
2	DEV2G5_2	1 Gbps	SERDES1G_2
3	DEV2G5_3	1 Gbps	SERDES1G_3
4 (NPI)	DEV2G5_4	1 Gbps	SERDES1G_4
5	DEV2G5_5	2.5 Gbps	SERDES6G_0
6	DEV2G5_6	2.5 Gbps	SERDES6G_1
11 (CPU0)	No port module	No macro	No macro
12 (CPU1)	No port module	No macro	No macro
13 (VD0)	No port module	No macro	No macro
14 (VD1)	No port module	No macro	No macro

Ports 0 and 1 are dual-media ports and support either a SerDes interface or a copper transceiver.

Ports 11 through 14 are internal ports defined as follows:

- Port 11 and port 12: CPU port 0 and 1 (CPU0, CPU1) are used for injection and extraction of frames.
- Port 13: Virtual device 0 (VD0) is used for multicast routing.
- Port 14: Virtual device 1 (VD1) is used for AFI frame injections.

The internal ports do not have associated port modules and interface macros.

The following table shows the fixed mapping from interface macros to the device I/O pins.

Table 5 • **Interface Macro to I/O Pin Mapping**

SERDES Macros	I/O Pin Names
CUPHY_0 to CUPHY_1	P0_D[3:0]N, P0_D[3:0]P through P1_D[3:0]N, P1_D[3:0]P
SERDES1G_0 to SERDES1G_4	S0_RXN, S0_RXP, S0_TXN, S0_TXP through S4_RXN, S4_RXP, S4_TXN, S4_TXP
SERDES6G_0 to SERDES6G_1	S5_RXN, S5_RXP, S5_TXN, S5_TXP through S6_RXN, S6_RXP, S6_TXN, S6_TXP

3.4.1 Supported SerDes Interfaces

The device supports a range of SerDes interfaces. The following table lists the SerDes interfaces supported by the SerDes ports, including standards, data rates, connectors, medium, and coding for each interface.

Table 6 • **Supported SerDes Interfaces**

Port Interface	Specification	Port Speed	Data Rate	Connector	Medium	Coding	SERDES1G	SERDES6G
100BASE-FX	IEEE 802.3, Clause 24	100M	125 Mbps	SFP	PCB	4B5B	x	x
SGMII	Cisco	1G	1.25 Gbps		PCB	8B10B	x	x
SFP	SFP-MSA	1G	1.25 Gbps	SFP	PCB	8B10B	x	x
1000BASE-KX	IEEE802.3, Clause 70	1G	1.25 Gbps		PCB, backplane	8B10B	x	x
2.5G	Proprietary (aligned to SFP)	2.5G	3.125 Gbps		PCB	8B10B		x

3.4.2 Dual-Media Mode

Ports 0 and 1 are dual-media ports and support either a 1G copper transceiver or a 1G SERDES interface. The dual-media mode is configurable per port through DEVCPU_GCB::PHY_CFG.PHY_ENA.

3.4.3 Logical Port Numbers

The analyzer and the rewriter uses in many places a logical port number. For instance, when link aggregation is enabled, all ports within a link aggregation group must be configured to use the physical port number of the port with the lowest port number within the group as logical port group ID. The mapping to a logical port number is configured in ANA_CL:PORT:PORT_ID_CFG.LPORT_NUM and REW::PORT_CTRL.ES0_LPORT_NUM.

3.5 SERDES1G

The SERDES1G is a high-speed SerDes interface that can operate at 1.25 Gbps (SGMII/SerDes, 1000BASE-KX). In addition, 100 Mbps (100BASE-FX) is supported by oversampling.

The SERDES1G supports the configuration of several parameters for increased performance in the specific application environment. The features include:

- Baudrate support 1.25 Gbps
- Programmable loop bandwidth and phase regulation behavior for clock and data recovery unit (CDR)
- Input buffer (IB) and output buffer (OB) configurations supporting:
  IB Signal Detect/Loss of Signal (LOS) options
  IB equalization (including corner frequency configuration)
  OB selectable de-emphasis
  OB amplitude drive levels
  OB slew rate control

3.6 SERDES6G

The SERDES6G is a high-speed SerDes interface, which can operate at the following data rates.

- 1.25 Gbps (SGMII/SerDes 1000BASE-KX)
- 2.5 Gbps
- 100 Mbps (100BASE-FX) is supported by oversampling.

The SERDES6G supports the configuration of several parameters for increased performance in the specific application environment. Features include:

- Configurable baud rate support from 1.25 Gbps to 2.5 Gbps.
- Programmable loop bandwidth and phase regulation behavior for clock and data recovery unit (CDR)
- Phase-synchronization of transmitter when used in multilane aggregates for low skew between lanes
- Input buffer (IB) and output buffer (OB) configurations supporting:
  IB signal detect and loss of signal (LOS) options
  IB adaptive equalization
  OB programmable de-emphasis, 3 taps
  OB amplitude drive levels
  OB slew rate control
- Loopbacks for system diagnostics

3.7 Copper Transceivers

This section describes the high-level functionality and operation of two built-in copper transceivers. The integration is kept as close to multichip PHY and switch designs as possible. This allows a fast path for software already running in a similar distributed design while still benefiting from the cost savings provided by the integration.

3.7.1 Register Access

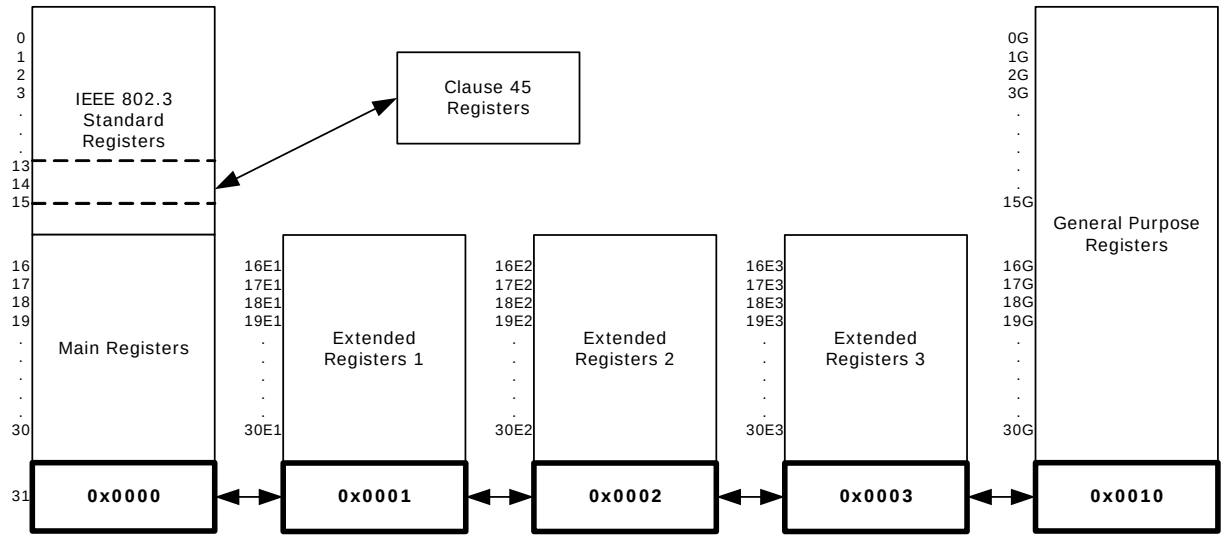
The registers of the integrated transceivers are not placed in the memory map of the switch, but are attached instead to the built-in MII management controller 0 of the device. As a result, PHY registers are accessed indirectly through the switch registers. For more information, see MII Management Controller, page 437.

In addition to providing the IEEE 802.3 specified 16 MII Standard Set registers, the PHYs contain an extended set of registers that provide additional functionality. The devices support the following types of registers:

- IEEE Clause 22 device registers with addresses from 0 to 31
- Two pages of extended registers with addresses from 16E1 through 30E1 and 16E2 through 30E2
- General-purpose registers with addresses from 0G to 30G
- IEEE Clause 45 device registers accessible through the Clause 22 registers 13 and 14 to support IEEE 802.3az Energy Efficient Ethernet registers

The memory mapping is controlled through PHY_MEMORY_PAGE_ACCESS::PAGE_ACCESS_CFG. The following illustration shows the relationship between the device registers and their address spaces.

Figure 17 • **Register Space Layout**



3.7.1.1 Broadcast Write

The PHYs can be configured to accept MII PHY register write operations, regardless of the destination address of these writes. This is enabled in PHY_CTRL_STAT_EXT::BROADCAST_WRITE_ENA. This enabling allows similar configurations to be sent quickly to multiple PHYs without having to do repeated MII PHY write operations. This feature applies only to writes; MII PHY register read operations are still interpreted with “correct” address.

3.7.1.2 Register Reset

The PHY can be reset through software, enabled in PHY_CTRL::SOFTWARE_RESET_ENA. Enabling this field initiates a software reset of the PHY. Fields that are not described as sticky are returned to their default values. Fields that are described as sticky are only returned to defaults if sticky-reset is disabled through PHY_CTRL_STAT_EXT::STICKY_RESET_ENA. Otherwise, they retain their values from prior to the software reset. A hardware reset always brings all PHY registers back to their default values.

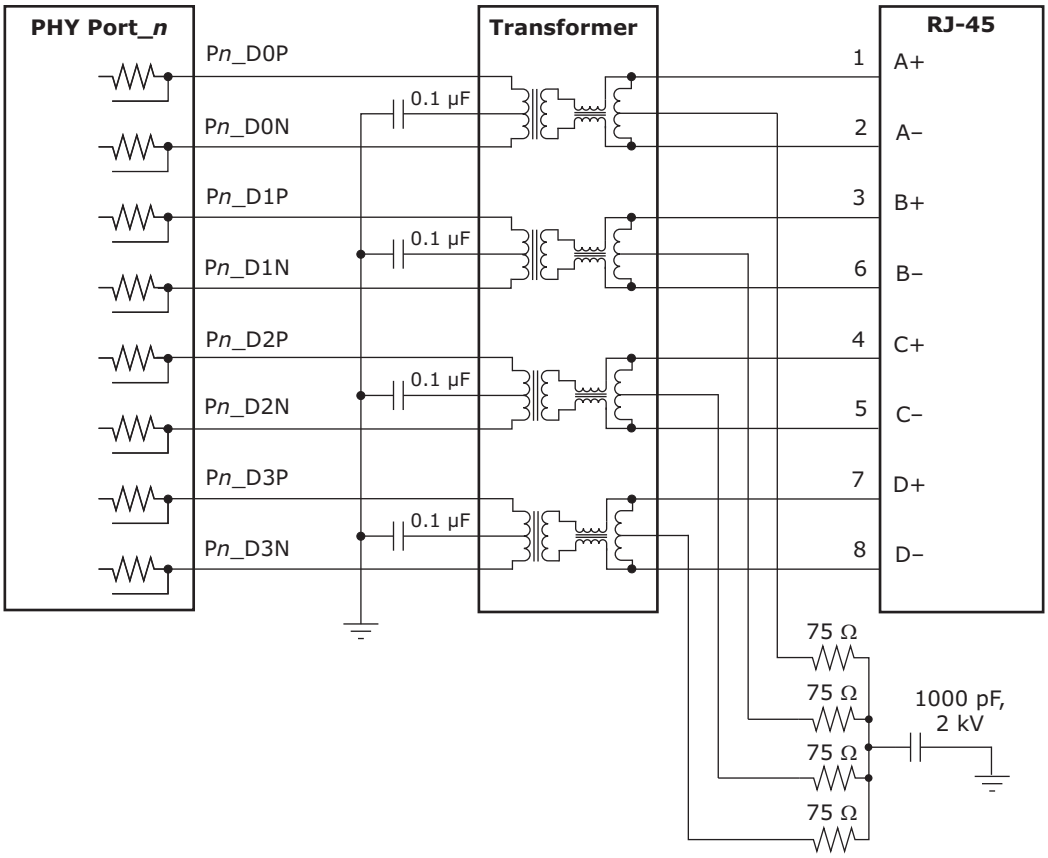
3.7.2 Cat5 Twisted Pair Media Interface

The twisted pair interfaces are compliant with IEEE 802.3-2008 and IEEE 802.3az for Energy Efficient Ethernet.

3.7.2.1 Voltage-Mode Line Driver

Unlike many other gigabit PHYs, this PHY uses a patented voltage-mode line driver that allows it to fully integrate the series termination resistors (required to connect the PHY’s Cat5 interface to an external 1:1 transformer). Also, the interface does not require placement of an external voltage on the center tap of the magnetic. The following illustration shows the connections.

Figure 18 • **Cat5 Media Interface**



3.7.2.2 Cat5 Auto-Negotiation and Parallel Detection

The integrated transceivers support twisted pair auto-negotiation as defined by clause 28 of the IEEE 802.3-2008. The auto-negotiation process evaluates the advertised capabilities of the local PHY and its link partner to determine the best possible operating mode. In particular, auto-negotiation can determine speed, duplex configuration, and master or slave operating modes for 1000BASE-TX. Auto-negotiation also allows the device to communicate with the link partner (through the optional “next pages”) to set attributes that may not otherwise be defined by the IEEE standard.

If the Cat5 link partner does not support auto-negotiation, the device automatically use parallel detection to select the appropriate link speed.

Auto-negotiation can be disabled by clearing PHY_CTRL.AUTONEG_ENA. If auto-negotiation is disabled, the state of the SPEED_SEL_MSB_CFG, SPEED_SEL_LSB_CFG, and DUPLEX_MODE_CFG fields in the PHY_CTRL register determine the device’s operating speed and duplex mode. Note that while 10BASE-T and 100BASE-T do not require auto-negotiation, 1000BASE-T does require it (defined by clause 40).

3.7.2.3 1000BASE-T Forced Mode Support

The integrated transceivers provides support for a 1000BASE-T forced test mode. In this mode, the PHY can be forced into 1000BASE-T mode and does not require manual setting of master/slave at the two ends of the link. This mode is only for test purposes. Do not use in normal operation. To configure a PHY in this mode, set PHY_EEE_CTRL.FORCE_1000BT_ENA = 1, with PHY_CTRL.SPEED_SEL_LSB_CFG = 1 and PHY_CTRL.SPEED_SEL_LSB_CFG = 0.

3.7.2.4 Automatic Crossover and Polarity Detection

For trouble-free configuration and management of Ethernet links, the integrated transceivers include a robust automatic crossover detection feature for all three speeds on the twisted-pair interface

(10BASE-T, 100BASE-T, and 1000BASE-T). Known as HP Auto-MDIX, the function is fully compliant with clause 40 of the IEEE 802.3-2002.

Additionally, the device detects and corrects polarity errors on all MDI pairs—a useful capability that exceeds the requirements of the standard.

Both HP Auto-MDIX detection and polarity correction are enabled in the device by default. Default settings can be changed using the POL_INV_DIS and PAIR_SWAP_DIS fields in the PHY_BYPASS_CTRL register. Status bits for each of these functions are located in the PHY_AUX_CTRL_STAT register.

The integrated transceivers can be configured to perform HP Auto-MDIX, even when auto-negotiation is disabled (PHY_CTRL.AUTONEG_ENA = 0) and the link is forced into 10/100 speeds. To enable the HP Auto-MDIX feature, set PHY_BYPASS_CTRL.FORCED_SPEED_AUTO_MDIX_DIS to 0.

The HP Auto-MDIX algorithm successfully detects, corrects, and operates with any of the MDI wiring pair combinations listed in the following table.

Table 7 • **Supported MDI Pair Combinations**

1, 2	3, 6	4, 5	7, 8	Mode
A	B	C	D	Normal MDI
B	A	D	C	Normal MDI-X
A	B	D	C	Normal MDI with pair swap on C and D pair
B	A	C	D	Normal MDI-X with pair swap on C and D pair

3.7.2.5 Manual MDI/MDI-X Setting

As an alternative to HP Auto-MDIX detection, the PHY can be forced to be MDI or MDI X using PHY_EXT_MODE_CTRL.FORCE_MDI_CROSSOVER_ENA. Setting this field to 10 forces MDI, and setting 11 forces MDI-X. Leaving the bits 00 enables the MDI/MDI-X setting to be based on FORCED_SPEED_AUTO_MDIX_DIS and PAIR_SWAP_DIS in the register PHY_BYPASS_CTRL.

3.7.2.6 Link Speed Downshift

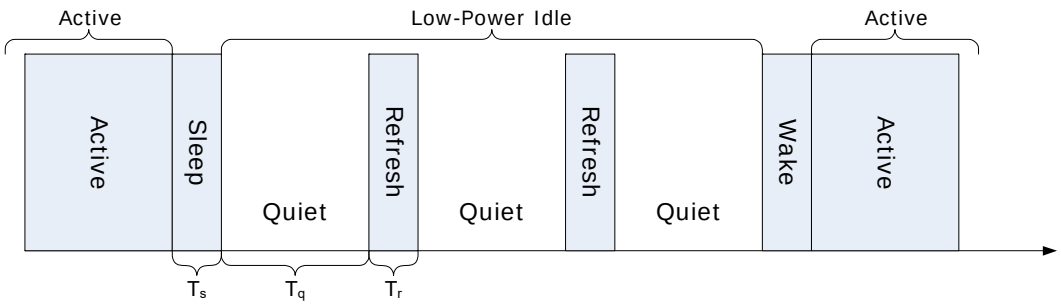
For operation in cabling environments that are incompatible with 1000BASE-T, the device provides an automatic link speed “downshift” option. When enabled, the device automatically changes its 1000BASE-T auto-negotiation advertisement to the next slower speed after a set number of failed attempts at 1000BASE-T. No reset is required to exit this state if a subsequent link partner with 1000BASE-T support is connected. This is useful in setting up in networks using older cable installations that may include only pairs A and B and not pairs C and D.

Link speed downshifting is configured and monitored using SPEED_DOWNSHIFT_STAT, SPEED_DOWNSHIFT_CFG, and SPEED_DOWNSHIFT_ENA in the register PHY_CTRL_EXT3.

3.7.2.7 Energy Efficient Ethernet

The integrated transceivers support IEEE 802.3az Energy Efficient Ethernet (EEE). This standard provides a method for reducing power consumption on an Ethernet link during times of low use.

Figure 19 • **Low Power Idle Operation**



Using LPI, the usage model for the link is to transmit data as fast as possible and then return to a low power idle state. Energy is saved on the link by cycling between active and low power idle states. Power is reduced during LPI by turning off unused circuits and, using this method, energy use scales with bandwidth utilization.

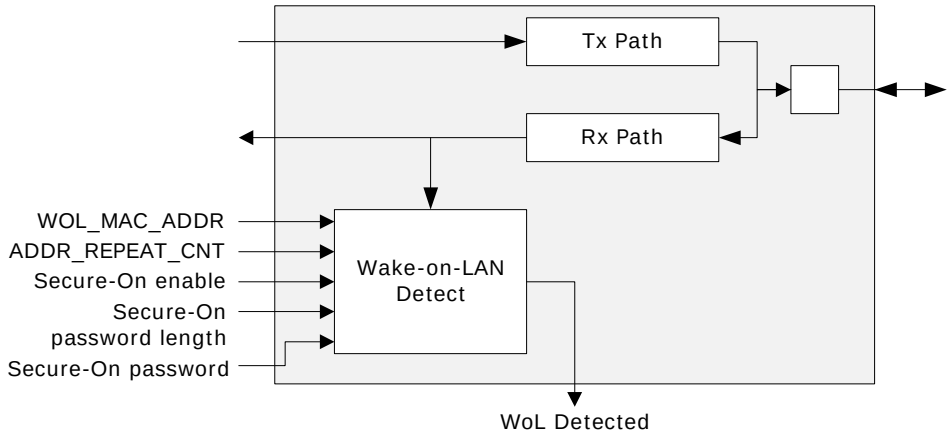
The transceivers use LPI to optimize power dissipation in 100BASE-TX and 1000BASE-T operation. In addition, IEEE 802.3az defines a 10BASE-Te mode that reduces transmit signal amplitude from 5 V to approximately 3.3 V, peak-to-peak. This mode reduces power consumption in 10 Mbps link speed and can fully interoperate with legacy 10BASE-T compliant PHYs over 100 m Cat5 cable or better.

To configure the transceivers in 10BASE-Te mode, set PHY_EEE_CTRL.EEE_LPI_RX_100BTX_DIS to 1 for each port. Additional Energy Efficient Ethernet features are controlled through Clause 45 registers as defined in Clause 45 registers to Support Energy Efficient Ethernet

3.7.3 Wake-On-LAN and SecureOn

The device supports Wake-on-LAN, an Ethernet networking standard to awaken hosts by using a “magic packet” that is decoded to ascertain the source, and then assert an interrupt pin or an LED. The device also supports SecureOn to secure Wake-on-LAN against unauthorized access. The following illustration shows an overview of the Wake-on-LAN functionality.

Figure 20 • **Wake-On-LAN Functionality**



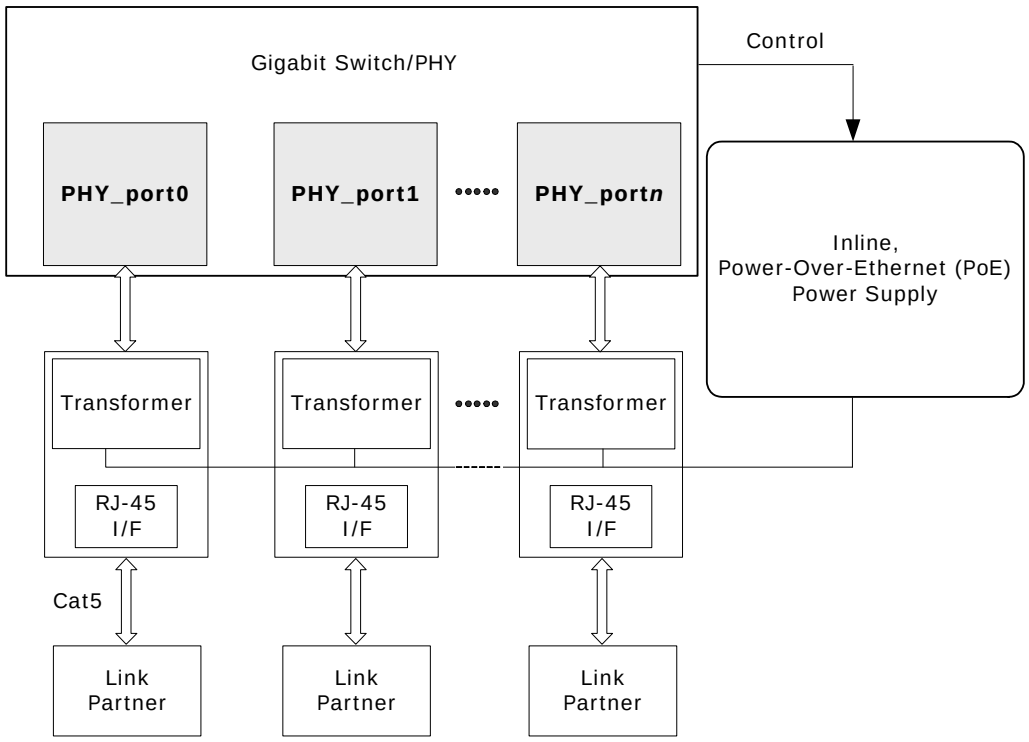
Wake-on-LAN detection is available in 10BASE-T, 100BASE-TX, and 1000BASE-T modes., It is enabled by setting the interrupt mask register PHY_INT_MASK. WOL_INT_ENA and its status is read in the interrupt status register PHY_INT_STAT. WOL_INT_PEND. Wake-on-LAN and SecureOn are configured for each port using the PHY_WOL_MDINT_CTRL register. The MAC address for each port is saved in its local register space (PHY_WOL_MAC_ADDRx).

3.7.4 Ethernet Inline Powered Devices

The integrated transceivers can detect legacy inline powered devices in Ethernet network applications. The inline powered detection capability can be part of a system that allows for IP-phone and other devices, such as wireless access points, to receive power directly from their Ethernet cable, similar to office digital phones receiving power from a Private Branch Exchange (PBX) office switch over the telephone cabling. This can eliminate the need of an external power supply for an IP-phone. It also enables the inline powered device to remain active during a power outage (assuming the Ethernet switch is connected to an uninterrupted power supply, battery, back-up power generator, or some other uninterruptable power source).

The following illustration shows an example of this type of application.

Figure 21 • **Inline Powered Ethernet Switch**



The following procedure describes the process that an Ethernet switch must perform to process inline power requests made by a link partner (LP); that is, in turn, capable of receiving inline power.

1. Enables the inline powered device detection mode on each transceiver using its serial management interface. Set PHY_CTRL_EXT4.INLINE_POW_DET_ENA to 1.
2. Ensures that the Auto-Negotiation Enable bit (register 0.12) is also set to 1. In the application, the device sends a special Fast Link Pulse (FLP) signal to the LP. Reading PHY_CTRL_EXT4.INLINE_POW_DET_STAT returns 00 during the search for devices that require Power-over-Ethernet (PoE).
3. The transceiver monitors its inputs for the FLP signal looped back by the LP. An LP capable of receiving PoE loops back the FLP pulses when the LP is in a powered down state. This is reported when PHY_CTRL_EXT4.INLINE_POW_DET_STAT reads back 01. If an LP device does not loop back the FLP after a specific time, PHY_CTRL_EXT4.INLINE_POW_DET_STAT automatically resets to 10.
4. If the transceiver reports that the LP needs PoE, the Ethernet switch must enable inline power on this port, externally of the PHY.
5. The PHY automatically disables inline powered device detection if PHY_CTRL_EXT4.INLINE_POW_DET_STAT automatically resets to 10, and then automatically changes to its normal auto-negotiation process. A link is then auto-negotiated and established when the link status bit is set (PHY_STAT.LINK_STAT is set to 1).
6. In the event of a link failure (indicated when PHY_STAT.LINK_STAT reads 0), the inline power must be disabled to the inline powered device external to the PHY. The transceiver disables its normal auto-negotiation process and re-enables its inline powered device detection mode.

3.7.5 IEEE 802.3af PoE Support

The integrated transceivers are also compatible with switch designs intended for use in systems that supply power to Data Terminal Equipment (DTE) by means of the MDI or twisted pair cable, as described in clause 33 of the IEEE 802.3af.

3.7.6 ActiPHY™ Power Management

In addition to the IEEE-specified power-down control bit (PHY_CTRL.POWER_DOWN_ENA), the device also includes an ActiPHY power management mode for each PHY. The ActiPHY mode enables support for power-sensitive applications. It uses a signal detect function that monitors the media interface for the presence of a link to determine when to automatically power-down the PHY. The PHY "wakes up" at a programmable interval and attempts to wake-up the link partner PHY by sending a burst of FLP over copper media.

The ActiPHY power management mode in the integrated transceivers is enabled on a per-port basis during normal operation at any time by setting PHY_AUX_CTRL_STAT.ACTIPHY_ENA to 1.

Three operating states are possible when ActiPHY mode is enabled:

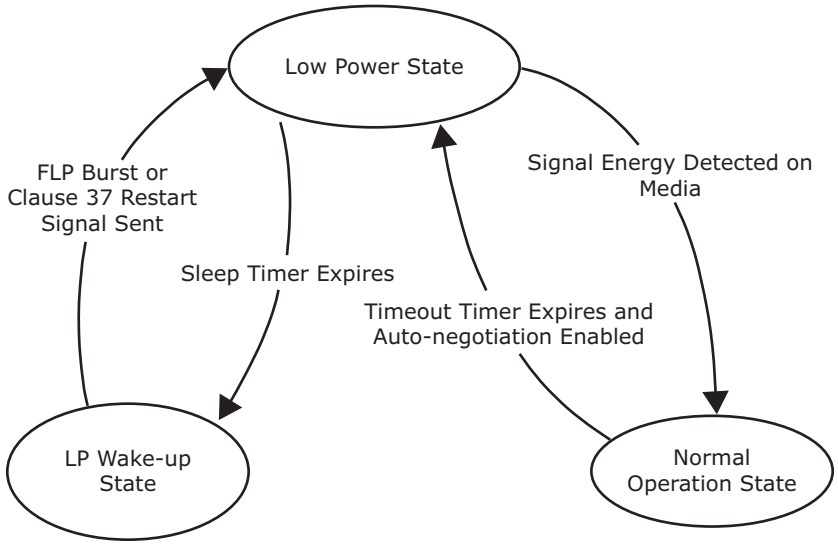
- Low power state
- LP wake-up state
- Normal operating state (link up state)

The PHY switches between the low power state and the LP wake-up state at a programmable rate (the default is two seconds) until signal energy is detected on the media interface pins. When signal energy is detected, the PHY enters the normal operating state. If the PHY is in its normal operating state and the link fails, the PHY returns to the low power state after the expiration of the link status time-out timer. After reset, the PHY enters the low power state.

When auto-negotiation is enabled in the PHY, the ActiPHY state machine operates as described. If auto-negotiation is disabled and the link is forced to use 10BT or 100BTX modes while the PHY is in its low power state, the PHY continues to transition between the low power and LP wake-up states until signal energy is detected on the media pins. At that time, the PHY transitions to the normal operating state and stays in that state even when the link is dropped. If auto-negotiation is disabled while the PHY is in the normal operation state, the PHY stays in that state when the link is dropped and does not transition back to the low power state.

The following illustration shows the relationship between ActiPHY states and timers.

Figure 22 • **ActiPHY State Diagram**



3.7.6.1 Low Power State

All major digital blocks are powered down in the lower power state.

In this state, the PHY monitors the media interface pins for signal energy. The PHY comes out of low power state and transitions to the normal operating state when signal energy is detected on the media. This happens when the PHY is connected to one of the following:

- Auto-negotiation capable link partner

- Another PHY in enhanced ActiPHY LP wake-up state

In the absence of signal energy on the media pins, the PHY transitions from the low power state to the LP wake-up state periodically based on the programmable sleep timer (PHY_CTRL_EXT3.ACTIPHY_SLEEP_TIMER). The actual sleep time duration is random, from –80 ms to 60 ms, to avoid two linked PHYs in ActiPHY mode entering a lock-up state during operation.

After sending signal energy on the relevant media, the PHY returns to the low power state.

3.7.6.2 Link Partner Wake-up State

In the link partner wake-up state, the PHY attempts to wake up the link partner. Up to three complete FLP bursts are sent on alternating pairs A and B of the Cat5 media for a duration based on the wake-up timer, which is set using register bits 20E1.12:11.

After sending signal energy on the relevant media, the PHY returns to the low power state.

3.7.6.3 Normal Operating State

In normal operation, the PHY establishes a link with a link partner. When the media is unplugged or the link partner is powered down, the PHY waits for the duration of the programmable link status time-out timer, which is set using ACTIPHY_LINK_TIMER_MSB_CFG and ACTIPHY_LINK_TIMER_LSB_CFG in the PHY_AUX_CTRL_STAT register. It then enters the low power state.

3.7.7 Testing Features

The integrated transceivers include several testing features designed to facilitate performing system-level debugging.

3.7.7.1 Ethernet Packet Generator (EPG)

The Ethernet Packet Generator (EPG) can be used at each of the 10/100/1000BASE-T speed settings for copper Cat5 media to isolate problems between the MAC and the PHY, or between a local PHY and its remote link partner. Enabling the EPG feature effectively disables all MAC interface transmit pins and selects the EPG as the source for all data transmitted onto the twisted pair interface.

Important The EPG is intended for use with laboratory or in-system testing equipment only. Do not use the EPG testing feature when the PHY is connected to a live network.

To use the EPG feature, set PHY_1000BT_EPG2.EPG_ENA to 1.

When PHY_1000BT_EPG2.EPG_RUN_ENA is set to 1, the PHY begins transmitting Ethernet packets based on the settings in the PHY_1000BT_EPG1 and PHY_1000BT_EPG2 registers. These registers set:

- Source and destination addresses for each packet
- Packet size
- Inter-packet gap
- FCS state
- Transmit duration
- Payload pattern

If PHY_1000BT_EPG1.TRANSMIT_DURATION_CFG is set to 0, PHY_1000BT_EPG1.EPG_RUN_ENA is cleared automatically after 30,000,000 packets are transmitted.

3.7.7.2 CRC Counters

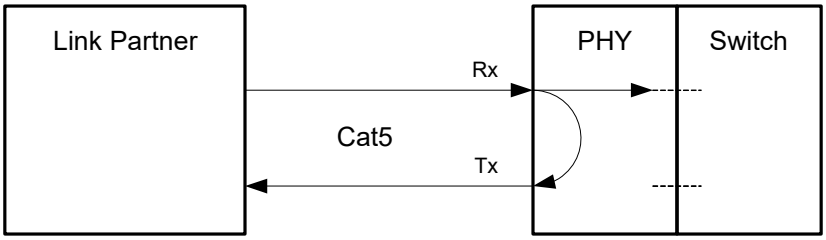
Two separate CRC counters are available in the PHY: a 14-bit good CRC counter available through PHY_CRC_GOOD_CNT.CRC_GOOD_PKT_CNT and a separate 8-bit bad CRC counter in PHY_CTRL_EXT4.CRC_1000BT_CNT.

3.7.7.3 Far-End Loopback

The far-end loopback testing feature is enabled by setting PHY_CTRL_EXT1.FAR_END_LOOPBACK_ENA to 1. When enabled, it forces incoming data from a link partner on the current media interface, into the MAC interface of the PHY, to be re-transmitted back to the link partner on the media interface as shown in the following illustration. The incoming data also appears

on the receive data pins of the MAC interface. Data present on the transmit data pins of the MAC interface is ignored when using this testing feature.

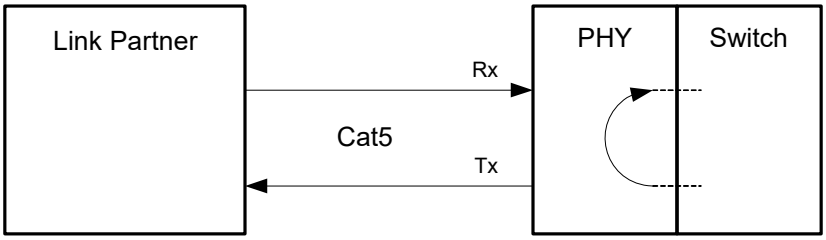
Figure 23 • **Far-End Loopback Diagram**



3.7.7.4 Near-End Loopback

When the near-end loopback testing feature is enabled (by setting PHY_CTRL.LOOPBACK_ENA to 1), data on the transmit data pins (TXD) is looped back in the PCS block, onto the device receive data pins (RXD), as shown in the following illustration. When using this testing feature, no data is transmitted over the network.

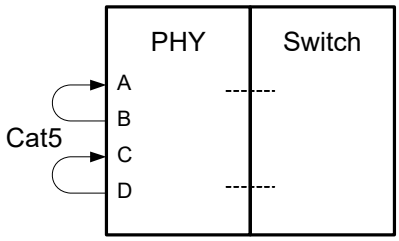
Figure 24 • **Near-End Loopback Diagram**



3.7.7.5 Connector Loopback

The connector loopback testing feature allows the twisted pair interface to be looped back externally. When using the connector loopback feature, the PHY must be connected to a loopback connector or a loopback cable. Pair A must be connected to pair B, and pair C to pair D, as shown in the following illustration. The connector loopback feature functions at all available interface speeds.

Figure 25 • **Connector Loopback Diagram**



When using the connector loopback testing feature, the device auto-negotiation, speed, and duplex configuration is set using device registers 0, 4, and 9. For 1000BASE-T connector loopback, the following additional writes are required, executed in the following steps:

1. Enable the 1000BASE-T connector loopback. Set PHY_CTRL_EXT2.CON_LOOPBACK_1000BT_ENA to 1.
2. Disable pair swap correction. Set PHY_CTRL_EXT2.CON_LOOPBACK_1000BT_ENA to 1.

3.7.8 VeriPHY™ Cable Diagnostics

The integrated transceivers include a comprehensive suite of cable diagnostic functions that are available through the on-board processor. These functions enable cable operating conditions and status to be accessed and checked. The VeriPHY suite has the ability to identify the cable length and operating conditions and to isolate common faults that can occur on the Cat5 twisted pair cabling.

For the functional details of the VeriPHY suite and the operating instructions, see the *ENT-AN0125, PHY, Integrated PHY-Switch VeriPHY - Cable Diagnostics Feature* Application Note.

3.8 DEV1G and DEV2G5 Port Modules

The DEV1G and DEV2G5 port modules are essentially the same with identical configurations. The only difference is the bandwidth of the bus connecting the port module to the assembler. A DEV2G5 port module can forward data at 2.5 Gbps while the DEV1G port module can forward data at 1 Gbps. The bandwidth of the port module is configured in the SERDES macro connecting to the port module.

3.8.1 MAC

This section describes the high level functions and configuration options of the Media Access controller (MAC) used in the DEV1G and DEV2G5 port modules.

The DEV1G MAC supports 10/100/1000 Mbps in full-duplex mode and 10/100 Mbps in half-duplex mode.

For the DEV2G5 MAC, the supported speeds and duplex modes depend on the following associated SERDES macro:

- SERDES1G: 10/100/1000 Mbps in full-duplex mode and 10/100 Mbps in half-duplex mode.
- SERDES6G: 10/100/1000/2500 Mbps in full-duplex mode and 10/100 Mbps in half-duplex mode.

The following table lists the registers associated with configuring the MAC.

Table 8 • **DEV1G and DEV2G5 MAC Configuration Registers Overview**

Register	Description	Replication
MAC_ENA_CFG	Enabling of Rx and Tx data paths	Per port module
MAC_MODE_CFG	Port mode configuration	Per port module
MAC_MAXLEN_CFG	Maximum length configuration	Per port module
MAC_TAGS_CFG	VLAN/service tag configuration	Per port module
MAC_TAGS_CFG2	VLAN/service tag configuration 2	Per port module
MAC_ADV_CHK_CFG	Configuration to enable dropping of Type/Length error frames	Per port module
MAC_IFG_CFG	Inter-frame gap configuration	Per port module
MAC_HDX_CFG	Half-duplex configuration	Per port module
MAC_STICKY	Sticky bit recordings	Per port module

3.8.1.1 Clock and Reset

There is a number of resets in the port module. All of the resets can be set and cleared simultaneously. By default, all blocks are in the reset state. With reference to DEV_RST_CTRL register, the resets are as listed in the following table.

Table 9 • **DEV1G and DEV2G5 Reset**

Register.Field	Description
DEV_RST_CTRL.MAC_RX_RST	Reset MAC receiver
DEV_RST_CTRL.MAC_TX_RST	Reset MAC transmitter
DEV_RST_CTRL.PCS_RX_RST	Reset PCS receiver
DEV_RST_CTRL.PCS_TX_RST	Reset PCS transmitter

When changing the MAC configuration, the port must go through a reset cycle. This is done by writing register DEV_RST_CTRL twice. On the first write, the reset bits are set. On the second write, the reset

bits are cleared. The non-reset field DEV_RST_CTRL.SPEED_SEL must keep its new value for both writes.

3.8.1.2 Interrupts

The following table lists the eight interrupt sources defined for DEV1G and DEV2G5 port modules. For more information about general interrupt handling, see Interrupt Controller, page 451.

Table 10 • **DEV1G and DEV2G5 Interrupt Sources Register Overview**

Register.Field	Description
DEV1G_INTR.FEF_FOUND_INTR_STICKY	Far-end-fault indication found
DEV1G_INTR.TX_LPI_INTR_STICKY	Low power idle transmit interrupt
DEV1G_INTR.RX_LPI_INTR_STICKY	Low power idle receive interrupt
DEV1G_INTR.AN_PAGE_RX_INTR_STICKY	New page event received by ANEG
DEV1G_INTR.AN_LINK_UP_INTR_STICKY	ANEG - Link status has changed to up
DEV1G_INTR.AN_LINK_DWN_INTR_STICKY	ANEG - Link status has changed to down
DEV1G_INTR.LINK_UP_INTR_STICKY	Link status is up
DEV1G_INTR. LINK_DWN_INTR_STICKY	Link status is down

3.8.1.3 Port Mode Configuration

The MAC provides a number of handles for configuring the port mode. With reference to the MAC_MODE_CFG, MAC_IFG_CFG, and MAC_ENA_CFG registers, the handles are as listed in the following table.

Table 11 • **DEV1G and DEV2G5 Port Mode Configuration Registers Overview**

Register.Field	Description
MAC_MODE_CFG.FDX_ENA	Enables full-duplex mode
MAC_ENA_CFG.RX_ENA	Enables MAC receiver module
MAC_ENA_CFG.TX_ENA	Enables MAC transmitter module
MAC_IFG_CFG.TX_IFG	Adjusts inter frame gap in Tx direction
DEV_RST_CTRL.SPEED_SEL	Configures port speed and data rate

Clearing MAC_ENA_CFG.RX_ENA stops the reception of frames and further frames are discarded. An ongoing frame reception is interrupted.

Clearing MAC_ENA_CFG.TX_ENA stops the dequeueing of frames from the egress queues, which means that frames are held back in the egress queues. An ongoing frame transmission is completed.

TX inter-frame gap (MAC_IFG_CFG.TX_IFG) must be set according to selected speed and mode to achieve 12 bytes IFG.

3.8.2 Half-Duplex Mode

The following special configuration options are available for half-duplex (HDX) mode.

- **Seed for back-off randomizer**. MAC_HDX_CFG.SEED seeds the randomizer used by the back-off algorithm. Use MAC_HDX_CFG.SEED_LOAD to load a new seed value.
- **Retransmission of frame after excessive collision**. MAC_HDX_CFG.RETRY_AFTER_EXC_COL_ENA determines whether the MAC retransmits frames after an excessive collision has occurred. If set, a frame is not dropped after excessive collisions, but the back-off sequence is restarted. This is a violation of IEEE 802.3, but is useful in non-dropping half-duplex flow control operation.

- **Late collision timing.** MAC_HDX_CFG.LATE_COL_POS adjusts the border between a collision and a late collision in steps of 1 byte. According to IEEE 802.3, section 21.3, this border is permitted to be on data byte 56 (counting frame data from 1); that is, a frame experiencing a collision on data byte 55 is always retransmitted, but it is never retransmitted when the collision is on byte 57. For each higher LATE_COL_POS value, the border is moved 1 byte higher.
- **Rx-to-Tx inter-frame gap**. The sum of MAC_IFG_CFG.RX_IFG1 and MAC_IFG_CFG.RX_IFG2 establishes the time for the Rx-to-Tx inter-frame gap. RX_IFG1 is the first part of half-duplex Rx-to-Tx inter-frame gap. Within RX_IFG1, this timing is restarted if carrier sense (CRS) has multiple high-low transitions (due to noise). RX_IFG2 is the second part of half-duplex Rx-to-Tx inter-frame gap. Within RX_IFG2, transitions on CRS are ignored.

3.8.2.1 Type/Length Check

The MAC supports frame lengths of up to 14,000 bytes. The maximum frame accepted by the MAC is configurable and defined in MAC_MAXLEN_CFG.MAX_LEN.

The MAC allows 1/2/3 tagged frames to be 4/8/12 bytes longer respectively, than the specified maximum length, with MAC_TAGS_CFG.VLAN_LEN_AWR_ENA. The MAC must be configured to look for VLAN tags (MAC_TAGS_CFG.VLAN_AWR_ENA). By default, EtherType 0x8100 and 0x88A8 are identified as VLAN tags. In addition, three custom EtherTypes can be configured by MAC_TAGS_CFG.TAG_ID, MAC_TAGS_CFG2.TAG_ID2, and MAC_TAGS_CFG2.TAG_ID3.

If a received frame exceeds the maximum length, the frame is truncated and marked as aborted.

The MAC can drop frames with in-range and out-of-range length errors by enabling MAC_ADV_CHK_CFG.LEN_DROP_ENA.

3.8.3 Physical Coding Sublayer (PCS)

This section provides information about the Physical Coding Sublayer (PCS) block, where the auto-negotiation process establishes mode of operation for a link. The PCS supports an SGMII mode and two SerDes modes, 1000BASE-X and 100BASE-FX.

The following table lists the registers associated with the PCS.

Table 12 • **DEV1G and DEV2G5 PCS Configuration Registers Overview**

Register	Description	Replication
PCS1G_CFG	PCS configuration	Per PCS
PCS1G_MODE_CFG	PCS mode configuration	Per PCS
PCS1G_SD_CFG	Signal Detect configuration	Per PCS
PCS1G_ANEG_CFG	Auto-negotiation configuration register	Per PCS
PCS1G_ANEG_NP_CFG	Auto-negotiation next page configuration	Per PCS
PCS1G_LB_CFG	Loopback configuration	Per PCS
PCS1G_ANEG_STATUS	Status signaling of PCS auto-negotiation process	Per PCS
PCS1G_ANEG_NP_STATUS	Status signaling of the PCS auto-negotiation next page process	Per PCS
PCS1G_LINK_STATUS	Link status	Per PCS
PCS1G_LINK_DOWN_CNT	Link down counter	Per PCS
PCS1G_STICKY	Sticky bit register	Per PCS

The PCS is enabled in PCS1G_CFG.PCS_ENA and PCS1G_MODE_CFG.SGMII_MODE_ENA selects between the SGMII and SerDes mode. For information about enabling 100BASE-FX, see 100BASE-FX.

The PCS supports the IEEE 802.3, Clause 66 unidirectional mode, where the transmission of data is independent of the state of the receive link (PCS1G_MODE_CFG.UNIDIR_MODE_ENA).

3.8.3.1 Auto-Negotiation

Auto-negotiation is enabled with PCS1G_ANEG_CFG.ANEG_ENA. To restart the auto-negotiation process, PCS1G_ANEG_CFG.ANEG_RESTART_ONE_SHOT must be set.

In SGMII mode (PCS1G_MODE_CFG.SGMII_MODE_ENA = 1), matching the duplex mode with the link partner must be ignored (PCS1G_ANEG_CFG.SW_RESOLVE_ENA). Otherwise the link is kept down when the auto-negotiation process fails.

The advertised word for the auto-negotiation process (base page) is configured in PCS1G_ANEG_CFG.ADV_ABILITY. The next page information is configured in PCS1G_ANEG_NP_CFG.NP_TX.

When the auto-negotiation state machine has exchanged base page abilities, the PCS1G_ANEG_STATUS.PAGE_RX_STICKY is asserted indicating that the link partner's abilities were received (PCS1G_ANEG_STATUS.LP_ADV_ABILITY).

If next page information need to be exchanged (only available in SerDes model, that is, PCS1G_MODE_CFG.SGMII_MODE_ENA = 0), PAGE_RX_STICKY must be cleared, next page abilities must be written to PCS1G_ANEG_NP_CFG.NP_TX, and PCS1G_ANEG_NP_CFG.NP_LOADED_ONE_SHOT must be set. When the auto-negotiation state machine has exchanged the next page abilities, the PCS1G_ANEG_STATUS.PAGE_RX_STICKY is asserted again indicating that the link partner's next page abilities were received (PCS1G_ANEG_STATUS.LP_NP_RX). Additional exchanges of next page information are possible using the same procedure.

Next page engagement is coded in bit 15 of Base Page of Next page information.

After the last next page has been received, the auto-negotiation state machine enters the IDLE_DETECT state, and the PCS1G_ANEG_STATUS.PR bit is set indicating that ability information exchange (base page and possible next pages) has finished and software can now resolve priority. Appropriate actions, such as Rx or Tx reset or auto-negotiation restart, can then be taken based on the negotiated abilities. The LINK_OK state is reached one link timer period later.

When the auto-negotiation process reached the LINK_OK state, PCS1G_ANEG_STATUS.ANEG_COMPLETE is asserted.

3.8.3.2 Link Surveillance

The current link status can be observed through PCS1G_LINK_STATUS.LINK_STATUS. The LINK_STATUS is defined as either the PCS synchronization state or as bit 15 of PCS1G_ANEG_STATUS.LP_ADV_ABILITY, which carries information about the link status in SGMII mode.

Link down is defined as the auto-negotiation state machine being in neither the AN_DISABLE_LINK_OK state nor the LINK_OK state for one link timer period. If a link down event occurred, PCS1G_STICKY.LINK_DOWN_STICKY is set and PCS1G_LINK_DOWN_CNT is incremented. In SGMII mode, the link timer period is 1.6 ms, whereas in SerDes mode, the link timer period is 10 ms.

The PCS synchronization state can be observed through PCS1G_LINK_STATUS.SYNC_STATUS. Synchronization is lost when the PCS is not able to recover and decode data received from the attached serial link.

3.8.3.3 Signal Detect

The PCS can be enabled to react to loss of signal through signal detect (PCS1G_SD_CFG.SD_ENA). Upon loss of signal, the PCS Rx state machine is restarted, and frame reception stops. If signal detect is disabled, no action is taken upon loss of signal. The polarity of signal detect is configurable in PCS1G_SD_CFG.SD_POL.

The source of signal detect is selected in PCS1G_SD_CFG.SD_SEL to either the SerDes PMA or the PMD receiver. If the SerDes PMA is used as source, the SerDes macro provides the signal detect. If the PMD receiver is used as source, signal detect is sampled externally through one of the GPIO pins on the device. For more information about the configuration of the GPIOs and signal detect, see Parallel Signal Detect, page 441.

PCS1G_LINK_STATUS.SIGNAL_DETECT contains the current value of the signal detect input.

3.8.3.4 Tx Loopback

For debug purposes, the Tx data path in the PCS can be looped back into the Rx data path. This is enabled through PCS1G_LB_CFG.TBI_HOST_LB_ENA.

3.8.3.5 Test Patterns

The following table lists the DEV1G and DEV2G5 registers associated with configuring test patterns.

Table 13 • **DEV1G and DEV2G5 PCS Test Pattern Configuration Registers**

Register	Description	Replication
PCS1G_TSTPAT_MODE_CFG	Test pattern configuration	Per PCS
PCS1G_TSTPAT_STATUS	Test pattern status	Per PCS

PCS1G_TSTPAT_MODE_CFG.JTP_SEL overwrites normal operation of the PCS and enables generation of jitter test patterns for debug. The jitter test patterns are defined in IEEE 802.3, Annex 36A. The following patterns are supported.

- High frequency test pattern
- Low frequency test pattern
- Mixed frequency test pattern
- Continuous random test pattern with long frames
- Continuous random test pattern with short frames

The PCS1G_TSTPAT_MODE_STATUS register holds information about error and lock conditions while running the jitter test patterns.

3.8.3.6 Low Power Idle

The following table lists the registers associated with Energy Efficient Ethernet (EEE) configuration and status in PCS.

Table 14 • **DEV1G and DEV2G5 PCS EEE Configuration Registers Overview**

Register	Description	Replication
PCS1G_LPI_CFG	Configuration of the PCS low power idle process	Per PCS
PCS1G_LPI_WAKE_ERROR_CNT	Wake error counter	Per PCS
PCS1G_LPI_STATUS	Low power idle status	Per PCS

The PCS supports Energy Efficient Ethernet (EEE) as defined by IEEE 802.3az. The PCS converts low power idle (LPI) encoding between the MAC and the serial interface transparently. In addition, the PCS provides control signals to stop data transmission in the SerDes macro. During low power idles, the serial transmitter in the SerDes macro can be powered down, only interrupted periodically while transmitting refresh information, which allows the receiver to notice that the link is still up but in power-down mode.

For more information about powering down the serial transmitter in the SerDes macro, see SERDES1G, page 34 or SERDES6G, page 35.

It is not necessary to enable the PCS for EEE, because it is controlled indirectly by the shared queue system. It is possible, however, to manually force the PCS into the low power idle mode through PCS1G_LPI_CFG.TX_ASSERT_LPIDLE. During LPI mode, the PCS constantly encodes low power idle with periodical refreshes. For more information about EEE, see Energy Efficient Ethernet, page 379.

The current low power idle state can be observed through PCS1G_LPI_STATUS for both receiver and transmitter:

- RX_LPI_MODE: Set if the receiver is in low power idle mode.
- RX_QUIET: Set if the receiver is in the quiet state of the low power idle mode. If cleared while RX_LPI_MODE is set, the receiver is in the refresh state of the low power idle mode.

The same is observable for the transmitter through TX_LPI_MODE and TX_QUIET.

If an LPI symbol is received, the RX_LPI_EVENT_STICKY bit is set, and if an LPI symbol is transmitted, the TX_LPI_EVENT_STICKY bit is set. These events are sticky.

The PCS1G_LPI_WAKE_ERROR_CNT wake-up error counter increments when the receiver detects a signal and the PCS is not synchronized. This can happen when the transmitter fails to observe the wake-up time or if the receiver is not able to synchronize in time.

3.8.3.7 100BASE-FX

The following table lists the registers associated with 100BASE-FX.

Table 15 • **DEV1G and DEV2G5 100BASE-FX Configuration Registers Overview**

Register	Description	Replication
PCS_FX100_CFG	Configuration of the PCS 100BASE-FX mode	Per PCS
PCS_FX100_STATUS	Status of the PCS 100BASE-FX mode	Per PCS

The PCS supports a 100BASE-FX mode in addition to the SGMII and 1000BASE-X SerDes modes. The 100BASE-X mode uses 4-bit/5-bit coding as specified in IEEE 802.3, Clause 24 for fiber connections. The 100BASE-FX mode is enabled through PCS_FX100_CFG.PCS_ENA, which masks out all PCS1G related registers.

The following options are available.

- Far-End Fault facility: In 100BASE-FX, the PCS supports the optional Far-End Fault facility. Both Far-End Fault Generation (PCS_FX100_CFG.FEFGEN_ENA) and Far-End Fault Detection (PCS_FX100_CFG.FEFCHK_ENA) are supported. A Far-End Fault incident is recorded in PCS_FX100_STATUS.FEF_FOUND.
- Signal Detect: 100BASE-FX has a similar signal detect scheme as of the SGMII and SERDES modes. For 100BASE-FX, PCS_FX100_CFG.SD_ENA enables signal detect, and PCS_FX100_CFG.SD_SEL selects the input source. The current status of the signal detect input can be observed through PCS_FX100_STATUS.SIGNAL_DETECT. For more information about signal detect, see Signal Detect, page 47.
- Link Surveillance: The PCS synchronization status can be observed through PCS_FX100_STATUS.SYNC_STATUS. When synchronization is lost the link breaks and PCS_FX100_STATUS.SYNC_LOST_STICKY is set. The PCS continuously tries to recover the link.
- Unidirectional mode: 100BASE-FX has a similar unidirectional mode as for the SGMII and SerDes modes, enabled through PCS_FX100_CFG.UNIDIR_MODE_ENA.

3.8.4 Port Statistics

Port statistics for DEV1G and DEV2G5 port modules are collected in the assembler and is accessed through registers in the assembler.

3.9 Assembler

The assembler (ASM) block is responsible for collecting words from the smaller taxi bus and assembling them into cells. It is also responsible for the loopback path between the rewriter and the analyzer.

For the first cell of a frame, which is the SOF cell, the assembler adds room for a 28-byte internal frame header (IFH). On a stacking port, the stacking tag is extracted from the frame data and placed in the respective part of the IFH.

The assembler receives a calendar sequence from the queue system defining in the sequence the ports should be served on the outgoing cell bus.

The assembler also detects PAUSE frames and forwards the extracted PAUSE information to the disassembler. PFC pause information extracted from PFC PAUSE frames are forwarded to the queue system.

Finally, the assembler collects the port statistics for all lower speed ports, which are the DEV1G and DEV2G5 ports. Statistics for the high-speed DEV10G ports are handled locally in the port module.

3.9.1 Setting Up a Port in the Assembler

The following table lists the port configuration registers within the assembler. Ethernet ports and CPU ports are configured using the same registers. Some of the fields are only relevant for setting up a CPU port (internal or external) and they will be covered in a later section.

Table 16 • **Port Configuration Register Overview**

Target::Register.Field	Description	Replication
ASM::PORT_CFG.NO_PREAMBLE_ENA	Preamble configuration of incoming frames.	Per port
ASM::PORT_CFG.SKIP_PREAMBLE_ENA	Preamble configuration of incoming frames.	Per port
ASM::PORT_CFG.PAD_ENA	Enable padding.	Per port
ASM::PORT_CFG.INJ_DISCARD_CFG	Configures discard behavior for injected frames.	Per port
ASM::PORT_CFG.INJ_FORMAT_CFG	Configure format of injected frames.	Per port
ASM::PORT_CFG.VSTAX2_AWR_ENA	Enable VStaX stacking header awareness.	Per port

By default, an Ethernet port does not need configuration in the assembler as long as special settings are not required. However, the following exceptions may apply:

If the port is used as a stacking port, set ASM::PORT_CFG.VSTAX2_AWR_ENA, which enables detection of the VStaX stacking header. If a VStaX stacking header is found, the assembler will remove the header from the frame and put it into the internal frame header.

Frames received from the port modules are preamble prepended by default. If a port module is configured for preamble shrink mode, the ASM::PORT_CFG.NO_PREAMBLE_ENA must be set to 1, because the port module does not prepend a preamble in this mode.

When ASM::PORT_CFG.PAD_ENA is set, frames that are smaller than 64 bytes are padded to reach the minimum frame size.

The assembler has a built-in frame fragment detection mechanism for incoming frames that were started but never received their EOF (because the port module was taken down, for example). These frames are normally finalized by creating an EOF abort marked cell. If a frame has been discarded, it is noted in ASM::PORT_STICKY.FRM_AGING_STICKY. The following table lists the port status register.

Table 17 • **Port Status Register Overview**

Target::Register.Field	Description	Replication
ASM::PORT_STICKY.IFH_PREFIX_ERR_STICKY	Injection format mismatch sticky bit	Per port
ASM::PORT_STICKY.FRM_AGING_STICKY	Frame aging sticky bit	Per port

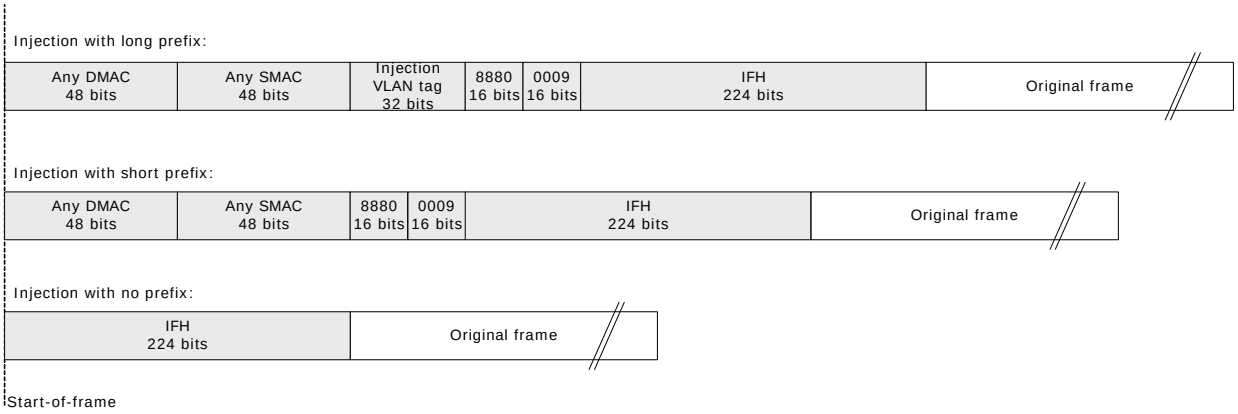
3.9.2 Setting Up a Port for Frame Injection

Any front port and the internal CPU ports (ports 11-12) can be configured for frame injection. Frames that are to be injected must have an IFH prepended the frame data. Optionally, the frame can further be prepended an SMAC, a DMAC, and a VLAN tag. This results in the following three prefix formats for injection frames:

- Injection with long prefix
- Injection with short prefix
- Injection with no prefix

The injection format is selected in ASM::PORT_CFG.INJ_FORMAT_CFG. The prefix formats are shown in the following illustration.

Figure 26 • **Frame Injection Formats**



For the long prefix, the incoming frame is matching against a programmable VLAN tag. The VLAN tag is common for all ports. Only the VID and TPID fields of the VLAN tag are compared. The following table lists the injection VLAN configuration register.

Table 18 • **Injection VLAN Register Overview**

Target::Register.Field	Description	Replication
ASM::INJ_VLAN_CFG.INJ_VID_CFG	Injection tag VID	1
ASM::INJ_VLAN_CFG.INJ_TPID_CFG	Injection tag TPID	1

When a port is setup to receive frames with either a short prefix or a long prefix, the incoming frames are checked against the selected prefix. If the prefix does not match, an error indication is set in ASM::PORT_STICKY.IFH_PREFIX_ERR_STICKY. Depending on the setting of ASM::PORT_CFG.INJ_DISCARD_CFG, the frames are processed in one of the three following modes.

- None. Both compliant and non-compliant frames are forwarded. Compliant frames are forwarded based on the IFH, and non-compliant frames are forwarded as regular frames.
- Drop non-compliant. Compliant frames are forwarded based on the IFH, and non-compliant frames are discarded.
- Drop compliant. Compliant frame are discarded, and non-compliant frames are forwarded as normal frames.

If a CPU port is used for frame injection, ASM::PORT_CFG.NO_PREAMBLE_ENA must be set to 1 so that the assembler does not expect frame data to be prepended with a preamble.

If a front port is used for frame injection, ASM::PORT_CFG.SKIP_PREAMBLE_ENA must be set to 1 to discard the preamble data before processing of the prepended injection header.

3.9.3 Setting Up MAC Control Sublayer PAUSE Frame Detection

The following table lists the PAUSE frame detection configuration registers in the assembler.

Table 19 • **PAUSE Frame Detection Configuration Register Overview**

Target::Register.Field	Description	Replication
ASM::PAUSE_CFG.ABORT_PAUSE_ENA	Stop forwarding of PAUSE frames	Per port
ASM::PAUSE_CFG.ABORT_CTRL_ENA	Stop forwarding of MAC control frames	Per port
ASM::MAC_ADDR_HIGH_CFG.MAC_ADDR_HIGH	Bits 47-24 of DMAC address checked in MAC_Control frames	Per port
ASM::MAC_ADDR_LOW_CFG.MAC_ADDR_LOW	Bits 23-0 of DMAC address checked in MAC_Control frames	Per port

The assembler is responsible for detecting PAUSE frames and forwarding the PAUSE value to the disassembler. Because PAUSE frames belong to the MAC control sublayer, they should in general be filtered and not forwarded to a higher layer; that is, forwarded on the cell bus so they reach the analyzer. The filtering is done by abort marking the frame.

The PAUSE frame and other MAC control frames can be forwarded to the analyzer. For PAUSE frame forwarding, ASM::PAUSE_CFG.ABORT_PAUSE_ENA must be set to 0. For other MAC control frame forwarding, ASM::PAUSE_CFG.ABORT_CTRL_ENA must be set to 0.

When PAUSE frames are received, the DMAC address is checked. The DMAC address must be either the 48-bit multicast address 01-80-C2-00-00-01 as mentioned by IEEE802.3 Annex 31B.1 or the MAC address of the port, which is configured in ASM::MAC_ADDR_HIGH_CFG.MAC_ADDR_HIGH and ASM::MAC_ADDR_LOW_CFG.MAC_ADDR_LOW.

Note: The values configured in ASM::MAC_ADDR_HIGH_CFG.MAC_ADDR_HIGH and ASM::MAC_ADDR_LOW_CFG.MAC_ADDR_LOW must match the value configured in DSM::MAC_ADDR_BASE_HIGH_CFG.MAC_ADDR_HIGH and DSM::MAC_ADDR_BASE_LOW_CFG.MAC_ADDR_LOW.

The number of received PAUSE frames is tracked as part of the port statistics. For more information, see Setting Up Assembler Port Statistics, page 52.

3.9.4 Setting Up PFC

The PFC module of the assembler identifies PFC PAUSE frames by checking if the DMAC matches the reserved multicast address 01-80-c2-00-00-01, the Type/Length field matches a MAC control frame (0x8808), and the opcode is indicating a PFC frame (0x0101). Upon receiving a PFC PAUSE frame, the assembler extracts and stores the timer information for all enabled priorities in the assembler PFC timers. The PFC module generates a stop signal towards the queue system based on the current timer values.

Detection of PFC frames is enabled by setting ASM::PFC_CFG.RX_PFC_ENA to 1 for the respective (port,priority). When enabling PFC the current link speed for the port must be configured in ASM::PFC_CFG.FC_LINK_SPEED for timing information to be evaluated correctly. The PFC configuration registers are shown in the following table.

Table 20 • **PFC Configuration Register Overview**

Target::Register.Field	Description	Replication
ASM::PFC_CFG.RX_PFC_ENA	Enable PFC per priority	Per port
ASM::PFC_CFG.FC_LINK_SPEED	Configure the link speed	Per port

3.9.5 Setting Up Assembler Port Statistics

The assembler collects port statistics for all DEV1Gs and DEV2G5s. Statistics are also collected from the disassembler and the assembler. Port statistics for DEV10Gs are not collected in the assembler and must be looked up in each DEV10G separately.

The following table lists the port statistics configuration register in the assembler.

Table 21 • **Port Statistics Configuration Register Overview**

Target::Register.Field	Description	Replication
ASM::STAT_CFG.STAT_CNT_CLR_SHOT	Statistics counter initialization.	1
ASM::PORT_CFG.CSC_STAT_DIS	Disables collection of statistics in the ASM.	Per port

Port statistics inside the ASM are stored in RAMs. Before they can be used, they must be initialized to 0 by setting ASM::STAT_CFG.STAT_CNT_CLR_SHOT to 1.

The statistic counters start counting as soon as the hardware has reset ASM::STAT_CFG.STAT_CNT_CLR_SHOT to 0.

For information about the lists of port statistics registers available per port, see ASM:DEV_STATISTICS register group in the Register List.

For the 10G capable ports, set ASM::PORT_CFG.CSC_STAT_DIS to 1 when the port is set up for speeds faster than 2.5G, because the collection of statistics are handled locally in the DEV10G. For lower speeds, the port uses a DEV2G5.

All statistic counters have a width of 32 bits, except for the five byte-counters; RX_IN_BYTES_CNT, RX_OK_BYTES_CNT, RX_BAD_BYTES_CNT, TX_OUT_BYTES_CNT, and TX_OK_BYTES_CNT. Those have an additional 4-bit MSB counter. When reading from counters with more than 32 bits, the non-MSB part has to be read first in order to latch the MSB part into a shadow register, where the value can be read afterward. For writing, the MSB part must be written first.

3.9.6 Setting Up the Loopback Path

Cells to be looped back by the assembler are received on a separate loopback cell bus interface from the rewriter. The assembler merges the loopback data with data received from the ports onto the outgoing cell bus towards the analyzer. Loopback data belongs to one of the three following types.

- Virtual device 0 (VD0)
- Virtual device 1 (VD1)
- Front port loopback (LBK)

Each of the three loopback traffic types has its own FIFO. VD0 is accessed at register replication 0, VD1 at register replication 1, and LBK at register replication 2. The following table lists the loopback configuration registers.

Table 22 • **Loopback FIFO Configuration Register Overview**

Target::Register.Field	Description	Replication
ASM::LBK_FIFO_CFG.FIFO_FLUSH	Flush all data in the FIFO.	Per FIFO
ASM::VD_FC_WM.VD_FC_WM	Watermark for flow control towards the queue system.	Per VD FIFO

A FIFO can be flushed using the ASM::LBK_FIFO_CFG.FIFO_FLUSH register. The register field clears itself when the flush is completed.

Cells in the FIFO can be discarded due to aging. If a frame is discarded due to aging, a per port sticky bit is set in ASM::LBK_AGING_STICKY. The following table lists the loopback FIFO sticky bit registers.

Table 23 • **Loopback FIFO Sticky-Bit Registers Overview**

Target::Register.Field	Description	Replication
ASM::LBK_AGING_STICKY.LBK_AGING_STICKY	Set if a frame has been discarded due to aging. Per port bitmask.	1
ASM::LBK_OVFLW_STICKY.LBK_OVFLW_STICKY	Set if a frame have been discarded due to overflow.	1

For VD0 and VD1, the loopback block pushes back towards the queue system to avoid FIFO overflows. The watermark for configuring the FIFO level at which push back is active can be set in ASM::VD_FC_WM.VD_FC_WM. The watermark can be configured individually for VD0 and VD1.

No flow control handling exists for the LBK FIFO. In case of overflow, all new cells received from the rewriter are discarded, and a bit in the ASM::LBK_OVFLW_STICKY.LBK_OVFLW_STICKY sticky-bit register is set.

3.10 Versatile Content-Aware Processor (VCAP)

The Versatile Content-Aware Processor (VCAP™) is a content-aware packet processor that allows wire-speed packet inspection for rich implementation of, for example, advanced VLAN and QoS classification and manipulations, IP source guarding, and security features for wireline and wireless applications. This is achieved by programming rules into the VCAP.

When a VCAP is enabled, every frame passing though the switch is analyzed and multiple keys are created based on the contents of the packet. The frame is examined to determine the frame type (for example, IPv4 TCP frame), so that the frame information is extracted according to the frame type, port-specific configuration, and classification results from the basic classification. Keys are applied to the VCAP and when there is a match between a key and a rule in the VCAP, the rule is then applied to the packet from which the key was extracted.

A rule consists of an entry, an action, and a counter. Keys are matched against the entry of a rule. When a rule is matched, the action is returned by the VCAP, and the rule's counter is incremented.

One and only one rule will match a key. If a key matches more than one rule, the rule at the highest address is selected. This means that a rule at high address takes priority over rules at lower addresses. When a key does not match a rule, a default rule is triggered. A default rule's action is programmable just like regular rules. Some VCAPs have more than one default rule; which default rule to use is determined at the same time as the key is created and applied to the VCAP.

This section provides information about entries and actions in general terms. When discussing an entry or an action, it is considered a vector of bits that is provided by software. For information about building classification matching (CLM) keys, see VCAP CLM Keys and Actions, page 70. For information about building ingress stage 2 (IS2) keys, see VCAP IS2 Keys and Actions, page 156. For information about building longest prefix matching (LPM) keys, see VCAP LPM: Keys and Action, page 137. For information about building egress stage 0 (ES0) keys, see VCAP_ES0 Lookup, page 323.

The following sections describe how to program rules into the VCAPs of the device. First, a general description of configuration and VCAP operations is provided, including a description of wide VCAP rules. Second, detailed information about individual VCAPs is provided. Finally, a few practical examples illustrate how everything fits together when programming rules into VCAPs.

3.10.1 Configuring VCAP

Registers are available in two targets: VCAP_ES0 and VCAP_SUPER. Use VCAP_ES0 to configure the VCAP ES0 target and the VCAP_SUPER target for all other VCAPs.

The following table lists the registers associated with VCAP_ES0 and VCAP_SUPER.

Table 24 • **VCAP_ES0 and VCAP_SUPER Registers**

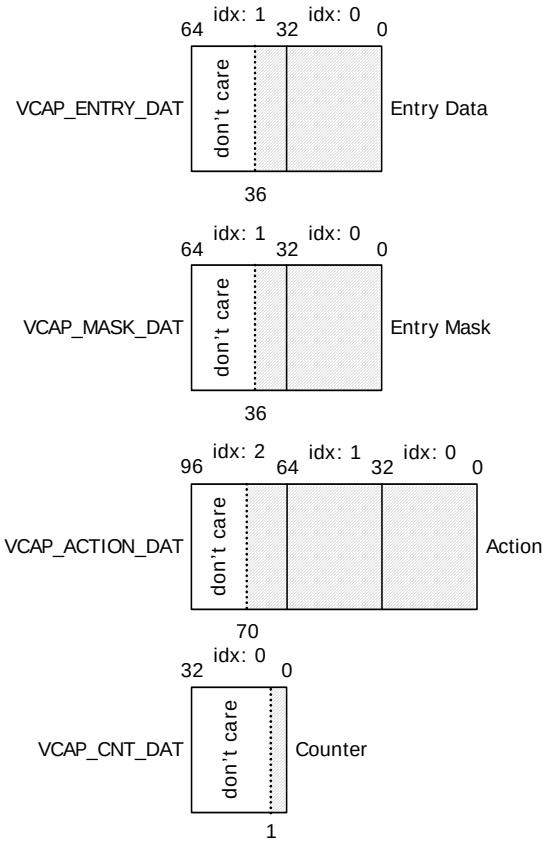
Register	Description
VCAP_UPDATE_CTRL	Initiates of read/write/move/initialization operations
VCAP_MV_CFG	Configures move/initialization operations
VCAP_CORE_IDX	Resource allocation, core index (does not apply to ES0)
VCAP_CORE_MAP	Resource allocation, mapping of cores (does not apply to ES0)
VCAP_ENTRY_DAT	Cache: Entry data
VCAP_MASK_DAT	Cache: Entry mask
VCAP_ACTION_DAT	Cache: Action
VCAP_CNT_DAT	Cache: Sticky-counter
VCAP_RULE_ENA	Cache: Rule enable (only applies to ES0)

A rule in the VCAP consists of an entry, action, and counter showing if the entry was hit. Rules are accessed indirectly though a cache. The cache corresponds to one address in VCAP memory. Software

access the cache using the VCAP configuration registers VCAP_ENTRY_DAT, VCAP_MASK_DAT, VCAP_ACTION_DAT, and VCAP_CNT_DAT.

The following illustration shows an example cache register mapping for a VCAP with 36-bit entry data/mask, 70-bit action, and a 1-bit sticky counter. Cache registers are replicated to accommodate fields that are wider than 32 bits. For example, if the entry size is 36 bits, bits [31:0] are accessed using VCAP_ENTRY_DAT[0][31:0], and bits [35:32] are accessed using VCAP_ENTRY_DAT[1][3:0].

Figure 27 • **VCAP Cache Layout Example**



Entries have both entry data and entry mask fields. This allows a don't-care of any bit position in the key when matching a key against entries.

Table 25 • **Entry Bit Encoding**

VCAP_ENTRY_DAT[*n*]/VCAP_MASK_DAT[*n*]	Description
0/0	Bit *n* is encoded for the match-0 operation. When a key is applied to the VCAP, this bit in the entry will match a 0 at position *n* in the key.
0/1	Bit *n* is encoded for match-any operation. When a key is applied to the VCAP, this bit in the entry will match both 0 and 1 at position *n* in the key. This encoding is sometimes referred to as don't care.
1/0	Bit *n* is encoded for match-1 operation. When a key is applied to the VCAP, this bit in the entry will match a 1 at position *n* in the key.
1/1	Bit *n* is encoded for match-off operation. The entry will never match any keys. This encoding is not available in the VCAP ES0. Instead, it is treated as match-any.

When programming a rule that does not use all the available bits in the cache, remaining (undefined) entry bits must be set to match-0, and action bits must be set to zeros.

To disable a rule, one or more bits in the entry is set to match-off. VCAP ES0 does not allow match-off encoding. Rules are instead enabled and disabled by the VCAP_RULE_ENA.RULE_ENA cache field.

The counter for a rule is incremented when the entry of the rule is matched by a key. Counters that are 1-bit wide do not wrap, but instead saturate at 1.

3.10.1.1 Writing, Reading, and Initializing Rules

All access to and from VCAP memory goes through the cache.

To write a rule in the VCAP memory at address a: Entry data, entry action, and counter is first written to the cache registers and then transferred into VCAP memory by triggering a write operation on the cache by setting VCAP_UPDATE_CTRL.UPDATE_CMD = 0, VCAP_UPDATE_CTRL.UPDATE_ADDR = a, and VCAP_UPDATE_CTRL.UPDATE_SHOT = 1. The UPDATE_SHOT field is cleared by hardware when the rule has put into VCAP memory.

To read a rule from address a in the VCAP memory: Trigger a read operation on the cache by setting VCAP_UPDATE_CTRL.UPDATE_CMD = 1, VCAP_UPDATE_CTRL.UPDATE_ADDR = a, and VCAP_UPDATA_CTRL.UPDATE_SHOT = 1. The UPDATE_SHOT field is cleared by hardware when a copy of the rule is available for reading via cache registers.

Active rules must not be overwritten by other rules (except when writing disabled rules). Software must make sure that addresses are initialized before they are written.

It is possible to write the contents of the cache can be written to a range of addresses inside VCAP memory. This is used during initialization of VCAP memory.

To write the contents of the cache to addresses a through b in VCAP memory, where a < b: Trigger an initialization-operation by setting VCAP_MV_CFG.MV_NUM_POS = b-a. And VCAP_UPDATE_CTRL.UPDATE_CMD = 4, VCAP_UPDATE_CTRL.UPDATE_ADDR = a, and VCAP_UPDATE_CTRL.UPDATA_SHOT = 1. When the UPDATE_SHOT field is cleared, the addresses have been overwritten with the contents of the cache.

The cache can be cleared by writing VCAP_UPDATE_CTRL.CLEAR_CACHE = 1. This immediately sets the contents of the entry to disabled, action and counter is set to all-zeros. VCAP_UPDATE_CTRL.CLEAR_CACHE can be set at the same time as triggering an write or initialization operation, the cache will be cleared before writing to the VCAP memory.

It is possible to selectively disable access to entry, action, and/or counter during operations by setting VCAP_UPDATE_CTRL.UPDATE_ENTRY_DIS, VCAP_UPDATE_CTRL.UPDATE_ACTION_DIS, or VCAP_UPDATE_CTRL.UPDATE_CNT_DIS. These fields allow specialized operations such as clearing counter values by performing initialization or write operations with disabled entry and action updating.

3.10.1.2 Moving a Block of Rules

The VCAP supports move operation for efficient and safe moving of rules inside VCAP memory. Moving may be required to make room for new rules, because rules are prioritized by address.

To move *n* addresses from address a to address b in VCAP memory: Trigger a move operation by setting VCAP_MV_CFG.MV_NUM_POS = (a>b ? a-b : b-a), VCAP_MV_CFG.MV_SIZE = (n-1). And setting VCAP_UPDATE_CTRL.UPDATE_CMD = (a>b ? 2 : 3), VCAP_UPDATE_CTRL.UPDATE_ADDR = a, and VCAP_UPDATE_CTRL.UPDATE_SHOT = 1. When the UPDATE_SHOT field is cleared, the contents of the addresses have been moved inside VCAP memory. The cache is overwritten during the move-operation.

Rules must not be moved to a region that contains active rules. Software must make sure that destination addresses are initialized before performing a move operation. After moving rules the move operation leaves VCAP memory in initialized state, so overlapping source and destination regions is not a problem during move operations.

3.10.1.3 Sharing Resources

The device implements a shared pool of six blocks that can be distributed freely among the CLM, IS2, and LPM VCAPs. Each block in the pool has 1,024 addresses with entries, actions, and counters.

Blocks are located back-to-back in the VCAP memory space. Block 0 is assigned addresses 0 through 1,023; block 1 is assigned addresses 1,024 though 2,047; and so on. Prioritizing of rules based on address still applies when multiple blocks are assigned to the same VCAP.

To simplify software development, all blocks assigned to the same VCAP should be allocated as one consecutive region of addresses in memory. After a block of VCAP resources is allocated to a VCAP, it cannot be reallocated. Resource assignment must be done during startup before the VCAPs are enabled.

After reset, all VCAP resources default to unallocated power-down mode. To allocate a block from the shared pool, write block index to VCAP_CORE_IDX.CORE_IDX, then map it to a specific interface by writing VCAP_CORE_MAP.CORE_MAP. For more information, see the register description for encoding of owners.

3.10.1.4 Bringing Up VCAP

The contents of the VCAP are unknown after reset. Software must initialize all addresses of the VCAP to disabled entry and all-zeros in action and counter before enabling lookups.

The default rules of the VCAPs must also be initialized. After lookup is enabled for the VCAPs, default rules are triggered and applied to frames.

Some VCAPs implement resource sharing. As a result, software must allocate resources as part of VCAP bring up.

For more information about default addresses rules and resource allocation, see Individual VCAPs, page 58.

3.10.2 Wide VCAP Entries and Actions

Some VCAPs support entries and actions that are wider than a single address in the VCAP memory. The size of an entry or action is described as X*n,* where *n* is the number of addresses that is taken up in memory. When programming an X2 or larger entry or action, addresses are written one by one until the entire entry or action is written to VCAP memory.

A rule consists of one entry and one action. The size of the rule is described as X*n* where *n* is the largest of entry or action widths. For example, a rule consisting of an X2 entry and an X4 action is considered to be an X4 rule. The entry and action must start at the same address, so that address offset 0 for both entry and action is located at the same address inside the VCAP memory.

When programming a rule where the action is wider than the entry, entry addresses exceeding the size of the entry must be disabled. When the entry is wider than the action, action addresses exceeding the size of the action must be set to zeros.

The starting address of an X*n* rule must be divisible by *n*. This means that X1 rules may be placed at any address, X2 rules may only placed at even addresses, X4 rules may only be placed at addresses divisible by 4, and so on. During move-operations this puts a restriction on the distance that rules are moved. In other words, when performing move-operation on a region of VCAP memory that contains rules of X2 or larger size, the rules must still be at legal addresses after the move.

When a rule is matched by a key, the counter at address offset 0 is incremented.

When writing an X2 or larger rule, software must start with the highest address and end with the lowest address. This is needed so that rules are not triggered prematurely during writing. When disabling rules, software must start by disabling the lowest address. This is automatically handled when using the initialization operation for disabling rules.

3.10.2.1 Type-Group Fields

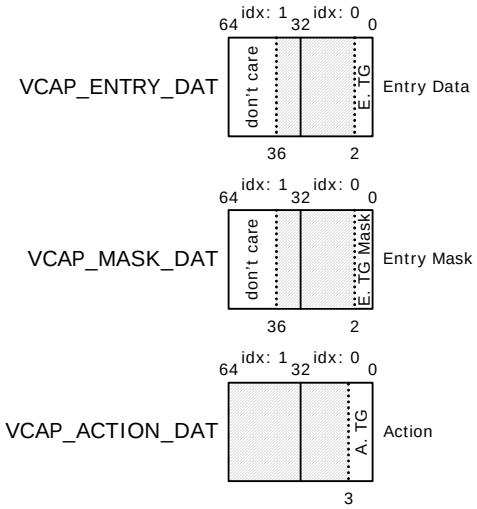
A special type-group field is required to differentiate rules when a VCAP supports different sized entries or actions. The type-group is not part of the entry/action description. Software has to manually lookup and insert type-group field into entry/action data when writing rules to VCAP memory.

The type-group field is placed at the low bits of the entry and/or action. The value and width of the type-group field differs between VCAPs and depends on the entry/action size of and the address offset into

the rule. For more information about type-group fields for individual VCAPs, see Individual VCAPs, page 58.

The following illustration shows an example of inserting a 2-bit entry type-group field and a 3-bit action type-group field for a VCAP with 36-bit entry data/mask and a 64-bit action. After inserting of type-group, there is room for 34 additional entry/mask bits and 61 action bits in the cache.

Figure 28 • **VCAP Cache Type-Group Example**



When programming the entry's type-group field, the mask bits must be set to zeros, so that the type-group bits consist of match-1 or match-0, or both.

Use the following procedure for each address that is part of an entry.

1. Use the entry's current address offset and size to find the appropriate type-group field and associated field-width by using the entry type-group table for the VCAP. Skip step 2 if entry type-group is not applicable for the VCAP.
2. Insert the entry type-group field into the cache for a type-group field width of *n*. Write the value of the type-group field to bit-positions [*n*-1:0] in VCAP_ENTRY_DAT and write zeros to bit-positions [*n*-1:0] in VCAP_MASK_DAT. In this way, type-group field value 1 becomes match-1 and 0 becomes match-0.
3. Fill the remainder of VCAP_ENTRY_DAT and VCAP_MASK_DAT with appropriate entry data. There will be room for (entry width – type-group width) additional bits of entry data.

Use the following procedure for each address is that is part of an action.

1. Use the action's current address offset and size to find the appropriate type-group field and associated field-width by using the action type-group table for the VCAP. Skip step 2 if action type-group is not applicable for the VCAP.
2. Insert the action type-group field into the cache. For a type-group field width of *n*. Write the value of the type-group field to bit positions [*n*-1:0] in VCAP_ACTION_DAT.
3. Fill the remainder of VCAP_ACTION_DAT with appropriate action data. There will be room for (action width – type-group width) additional bits of action data.

Counters never use type-group encoding.

3.10.3 Individual VCAPs

This section provides detailed information about the individual VCAPs; VCAP CLM, VCAP IS2, VCAP LPM, and VCAP ES0. The CLM, IS2, and LPM VCAPs share resources. For more information, see Sharing Resources, page 56.

3.10.3.1 VCAP CLM

The following table lists the parameters that apply to the VCAP CLM.

Table 26 • **VCAP CLM Parameters**

Parameter	Description
Number of VCAP addresses	1,024 per block that is allocated to this VCAP
Width of entry	36 bits per address
Width of action	70 bits per address
Counter type	1 bit saturating counter
Supported entry sizes	X1, X2, X4, X8, and X16
Supported action sizes	X1, X2, X4
Associated register target	VCAP_SUPER

The type-group field must be inserted into entries when programming rules, because the VCAP CLM supports more than one entry size.

Table 27 • **VCAP CLM Entry Type-Group Fields**

Address Offset	Size	Entry Type Group	Description
0	X1	0b1	Software must insert 1 bit with the value 1 into first address of all X1 entries.
0	X2	0b10	Software must insert 2 bit with the value 2 into first address of all X2 entries.
0	X4	0b100	Software must insert 3 bit with the value 4 into first address of all X4 entries.
0	X8	0b1000	Software must insert 4 bit with the value 8 into first address of all X8 entries.
0	X16	0b10000	Software must insert 5 bit with the value 16 into first address of all X16 entries.
1, 3, 5, 7, 9, 11, 13, and 15		0b0	Software must insert 1 bit with the value 0 into uneven addresses of all entries.
2, 6, 10, and 14		0b00	Software must insert 2 bit with the value 0 into addresses 2, 6, 10, and 14 of all entries.
4 and 12		0b000	Software must insert 3 bit with the value 0 into addresses 4 and 12 of all entries.
8		0b0000	Software must insert 4 bit with the value 0 into addresses 8 of all entries.

The type-group field must be inserted into actions when programming rules, because the VCAP CLM supports more than one action size.

Table 28 • **VCAP CLM Action Type-Group Fields**

Address Offset	Size	Action Type Group	Description
0	X1	0b1	Software must insert 1 bit with the value 1 into first address of all X1 actions.
0	X2	0b10	Software must insert 2 bit with the value 2 into first address of all X2 actions.

Table 28 • **VCAP CLM Action Type-Group Fields *(continued)***

Address Offset	Size	Action Type Group	Description
0	X4	0b100	Software must insert 3 bit with the value 4 into first address of all X4 actions.
1 and 3		0b0	Software must insert 1 bit with the value 0 into uneven addresses of all actions.
2		0b00	Software must insert 2 bit with the value 0 into address 2 of all actions.

It is possible to calculate the distribution of entry and action bits in the VCAP CLM based on entry and action widths, together with the type-group field-widths.

Table 29 • **VCAP CLM Entry/Action Bit Distribution**

Address offset	X1	X2	X4	X8	X16
0	e[34:0], a[68:0]	e[33:0], a[67:0]	e[32:0], a[66:0]	e[31:0]	e[30:0]
1		e[68:34], a[136:68]	e[67:33], a[135:67]	e[66:32]	e[65:31]
2			e[101:68], a[203:136]	e[100:67]	e[99:66]
3			e[136:102], a[272:204]	e[135:101]	e[134:100]
4				e[168:136]	e[167:135]
5				e[203:169]	e[202:168]
6				e[237:204]	e[236:203]
7				e[272:238]	e[271:237]
8					e[303:272]
9					e[338:304]
10					e[372:339]
11					e[407:373]
12					e[440:408]
13					e[475:441]
14					e[509:476]
15					e[544:510]

The VCAP CLM implements default rules. When submitting a key to the VCAP CLM, the analyzer simultaneously decides which default rule to hit if the key does not match any entries in the VCAP. If no resources (blocks) are assigned to VCAP CLM, there can be no matches. As a result, default rules will be hit.

Table 30 • **VCAP CLM Default Rule Addresses**

VCAP	Number of Default Rules	Address
CLM0	30	Starting address of CLM0 default rule number *n* is (6,144 + *n* × 16).
CLM1	30	Starting address of CLM1 default rule number *n* is (6,624 + *n* × 16).
CLM2	30	Starting address of CLM2 default rule number *n* is (7,104 + *n* × 16).

3.10.3.2 VCAP IS2

The following table lists the parameters that apply to the VCAP IS2.

Table 31 • **VCAP IS2 Parameters**

Parameter	Description
Number of VCAP addresses	1,024 per block that is allocated to this VCAP
Width of entry	36 bits per address
Width of action	70 bits per address
Counter-type	1-bit saturating counter
Supported entry sizes	X4, X8, X16
Supported action sizes	X4
Associated register target	VCAP_SUPER

The type-group field must be inserted into entries when programming rules, because the VCAP IS2 supports more than one entry size.

Table 32 • **VCAP IS2 Entry Type-Group Fields**

Address Offset	Size	Entry Type-Group	Description
0	X4	0b1	Software must insert 1 bit with the value 1 into first address of all X4 entries.
0	X8	0b10	Software must insert 2 bit with the value 2 into first address of all X8 entries.
0	X16	0b100	Software must insert 3 bit with the value 4 into first address of all X16 entries.
1, 2, 3, 5, 6, 7, 9, 10, 11, 13, 14, and 15			Software must not insert type-group into addresses 1, 2, 3, 5, 6, 7, 9, 10, 11, 13, 14, and 15 for entries.
4 and 12		0b0	Software must insert 1 bit with the value 0 into addresses 4 and 12 of all entries.
8		0b00	Software must insert 2 bits with the value 0 into address 8.

Because only one action size is supported by the VCAP IS2, no type-group is inserted into actions when programming rules.

It is possible to calculate the distribution of entry and action bits in the VCAP IS2 based on entry and action widths together with the entry type-group field width.

Table 33 • **VCAP IS2 Entry/Action Bit Distribution**

Address Offset	X4	X8	X16
0	e[34:0], a[69:0]	e[33:0]	e[32:0]
1	e[70:35], a[139:70]	e[69:34]	e[68:33]
2	e[106:71], a[209:140]	e[105:70]	e[104:69]
3	e[142:107], a[279:210]	e[141:106]	e[140:105]
4		e[176:142]	e[175:141]

Table 33 • **VCAP IS2 Entry/Action Bit Distribution *(continued)***

Address Offset	X4	X8	X16
5		e[212:177]	e[211:176]
6		e[248:213]	e[247:212]
7		e[284:249]	e[283:248]
8			e[317:284]
9			e[353:318]
10			e[389:354]
11			e[425:390]
12			e[460:426]
13			e[496:461]
14			e[532:497]
15			e[568:533]

The VCAP IS2 has 16 default rules. The address of default rule number *n* is (7584 + 16 × *n*). When submitting a key to the VCAP IS2, the analyzer simultaneously determines which default rule to hit if the key does not match any entries. If no resources (blocks) were assigned to VCAP IS2, there can be no matches and default rules will be hit.

3.10.3.3 VCAP LPM

The following table lists the parameters that apply to the VCAP LPM.

Table 34 • **VCAP LPM Parameters**

Parameter	Description
Number of VCAP addresses	1,024 per block that is allocated to this VCAP
Width of entry	36 bit per address
Width of action	70 bit per address
Counter-type	1 bit saturating counter
Supported entry sizes	X1, X2, X4, X8
Supported action sizes	X1
Associated register target	VCAP_SUPER

The type-group field must be inserted into entries when programming rules, because the VCAP LPM supports more than one entry size.

Table 35 • **VCAP LPM Entry Type-Group Fields**

Address Offset	Size	Entry Type Group	Description
0	X1	0b1	Software must insert 1 bit with the value 1 into first address of all X1 entries.
0	X2	0b10	Software must insert 2 bit with the value 2 into first address of all X2 entries.
0	X4	0b100	Software must insert 3 bit with the value 4 into first address of all X4 entries.
0	X8	0b1000	Software must insert 4 bit with the value 8 into first address of all X8 entries.

Table 35 • **VCAP LPM Entry Type-Group Fields *(continued)***

Address Offset	Size	Entry Type Group	Description
1, 3, 5, and 7		0b0	Software must insert 1 bit with the value 0 into uneven addresses of all entries.
2 and 6		0b00	Software must insert 2 bit with the value 0 into addresses 2 and 6 of all entries.
4		0b000	Software must insert 3 bit with the value 0 into addresses 4 of all entries.

Because only one action size is supported by the VCAP LPM, no type-group is inserted into actions when programming rules.

It is possible to calculate the distribution of entry and action bits in the VCAP LPM based on entry and action widths together with the entry type-group field width.

Table 36 • **VCAP LPM Entry/Action Bit Distribution**

Address Offset	X1	X2	X4	X8
0	e[34:0], a[69:0]	e[33:0]	e[32:0]	e[31:0]
1		e[68:34]	e[67:33]	e[66:32]
2			e[101:68]	e[100:67]
3			e[136:102]	e[135:101]
4				e[168:136]
5				e[203:169]
6				e[237:204]
7				e[272:238]

The VCAP LPM does not implement default rules. If the key does not match any entries, or if no resources (blocks) are assigned to the VCAP LPM, routing lookups are treated as misses.

3.10.3.4 VCAP ES0

VCAP ES0 has 512 addresses available. The following parameters apply to the VCAP ES0.

Table 37 • **VCAP ES0 Parameters**

Parameter	Description
Number of VCAP addresses	512.
Width of entry	30 bits per address.
Width of action	285 bits per address.
Counter-type	1-bit saturating counter.
Supported entry sizes	X1.
Supported action sizes	X1.
Associated register target	VCAP_ES0.

The type-group field is not used when programming entries and actions, because VCAP ES0 supports only one entry size and one action size.

The following table shows the distribution of entry and action bits in the VCAP ES0.

Table 38 • **VCAP ES0 Entry/Action Bit Distribution**

Address Offset	X1
0	e[29:0], a[284:0]

The VCAP ES0 has 11 default rules. The address of default rule number *n* is (512 + *n*). When submitting a key to the VCAP ES0, the rewriter simultaneously decides which default rule to hit if the key does not match any entries in the VCAP.

3.10.4 VCAP Programming Examples

This section provides examples of programming VCAP CLM and VCAP ES0.

3.10.4.1 Writing a Wide CLM Rule

This example shows how to write a CLM X4 rule, consisting of an X2 TRI_VID entry and an X4 FULL action, to the CLM1 VCAP. In this example, the second and third blocks of VCAP resources have already been mapped to CLM1.

When the second and third resource blocks are mapped to CLM1, it then owns VCAP address range 1024-3071. An X4 rule must be placed on an address inside memory owned by CLM1 and starting addresses must be divisible by four, because it is an X4 rule. In this example, the X4 rule is written to addresses 1412-1415.

Software is expected to track memory usage so that it only writes to VCAP memory that does not already contain active rules. Deleting of rules is done by initializing the associated addresses.

The X2 TRI_VID entry is 69 bits wide. The X4 FULL action is 250 bits wide. For information about how to build a FULL action, see VCAP CLM Actions, page 91. For information about how to build a TRI_VID entry, see VCAP CLM X2 Key Details, page 74.

Each address holds 36 bits of entry data/mask, but some of them are used for type-group fields. For more information about TRI_VID's entry type-group field and entry bit-ranges for address offset 0 and 1, Table 27, page 59 and Table 29, page 60. The results are shown in the following table.

Table 39 • **CLM X2 Entry Type-Group and Distribution Summary**

Address	Type-Group Field	Entry Data/Mask Range
1412	0b10	[33:0]
1413	0b0	[68:34]

Note: The TRI_VID entry completely fills available entry space. If it had been less than 69 bits wide, then MSBs at address offset 1 would have been treated as Match-0.

Each address holds 70 bits of action, but some of them are used for type-group fields. For more information the FULL's action type-group field and the action bit ranges for all four address offsets, see Table 28, page 59 and Table 29, page 60. The results are shown in the following table.

Table 40 • **CLM X4 Action Type-Group and Distribution Summary**

Address	Type-Group Field	Action Data/Mask Range
1412	0b100	[66:0]
1413	0b0	[135:67]
1414	0b00	[203:136]
1415	0b0	[249:204]

The FULL action is only 250 bits wide, so bits [272:250] of address offset 3 must be treated as zeros.

Software must write the highest address offset first and work down to the lowest address offset.

3.10.4.1.1 Writing Address Offset 3 of X4 Rule

Software is not allowed to modify the cache while VCAP operation is ongoing. Always check if VCAP_UPDATE_CTRL.UPDATE_SHOT is cleared before starting to modify the cache. If a previous operation is still ongoing, wait for the UPDATE_SHOT to clear.

When a new rule is started, first clear cache's entry, action, and counter by setting VCAP_UPDATE_CTRL.CLEAR_CACHE = 1.

The TRI_VID entry is an X2 entry and not part of address offset 3. The entry at this address must be set to Match-off. This has already been achieved by clearing of the cache.

The FULL action is not so wide that it needs a third VCAP_ACTION_DAT replication so this must be set to zero. This has already been achieved by clearing of the cache.

Write action to cache: VCAP_ACTION_DAT[1][31:0] = zero-extend(action[249:235]), VCAP_ACTION_DAT[0][31:1] = action[234:204], and VCAP_ACTION_DAT[0][0] = 0, because type-group field for the X4 CLM action at address offset 3 is 0b0.

Write cache to VCAP memory by setting VCAP_UPDATE_CTRL = (4|(1415<<3)).

3.10.4.1.2 Writing Address Offset 2 of X4 Rule

Wait for VCAP_UPDATE_CTRL.UPDATE_SHOT to clear.

The TRI_VID entry is an X2 entry and is not part of address offset 2, the entry at this address must be set to Match-off. This has already been achieved by clearing of the cache during write of address offset 3.

Write action to cache: VCAP_ACTION_DAT[2][31:0] = zero-extend(action[203:198]), VCAP_ACTION_DAT[1][31:0] = action[197:166], VCAP_ACTION_DAT[0][31:2] = action[165:136], and VCAP_ACTION_DAT[0][1:0] = 0, because type-group field for the X4 CLM action at address offset 2 is 0b00.

Write cache to VCAP memory by setting VCAP_UPDATE_CTRL = (4|(1414<<3)).

3.10.4.1.3 Writing Address Offset 1 of X4 Rule

Wait for VCAP_UPDATE_CTRL.UPDATE_SHOT to clear.

Write entry data to cache: VCAP_ENTRY_DAT[1][31:0] = zero-extend(entry data[68:65]), VCAP_ENTRY_DAT[0][31:1] = entry data[64:34], and VCAP_ENTRY_DAT[0][0] = 0, because type-group field for the X2 CLM entry at address offset 1 is 0b0.

Write entry mask to cache: VCAP_MASK_DAT[1][31:0] = zero-extend(entry mask[68:65]), VCAP_MASK_DAT[0][31:1] = entry mask[64:34], and VCAP_MASK_DAT[0][0] = 0, because type-group field must not be don't-cared.

Write action to cache: VCAP_ACTION_DAT[2][31:0] = zero-extend(action[135:130]), VCAP_ACTION_DAT[1][31:0] = action[129:98], VCAP_ACTION_DAT[0][31:1] = action[97:67], and VCAP_ACTION_DAT[0][0] = 0, because type-group field for the X4 CLM action at address offset 1 is 0b0.

Write cache to VCAP memory by setting VCAP_UPDATE_CTRL = (4|(1413<<3)).

3.10.4.1.4 Writing Address Offset 0 of X4 Rule

Wait for VCAP_UPDATE_CTRL.UPDATE_SHOT to clear.

Write entry data to cache: VCAP_ENTRY_DAT[1][31:0] = zero-extend(entry data[33:30]), VCAP_ENTRY_DAT[0][31:2] = entry data[29:0], and VCAP_ENTRY_DAT[0][1:0] = 2, because type-group field for the X2 CLM entry at address offset 0 is 0b10.

Write entry mask to cache: VCAP_MASK_DAT[1][31:0] = zero-extend(entry mask[33:30]), VCAP_MASK_DAT[0][31:2] = entry mask[29:0], and VCAP_MASK_DAT[0][1:0] = 0, because type-group field must not be don't-cared.

Write action to cache: VCAP_ACTION_DAT[2][31:0] = zero-extend(action[66:61]), VCAP_ACTION_DAT[1][31:0] = action[60:29], VCAP_ACTION_DAT[0][31:3] = action[28:0], and VCAP_ACTION_DAT[0][2:0] = 4, because type-group field for the X4 CLM action at address offset 0 is 0b100.

Write cache to VCAP memory by setting VCAP_UPDATE_CTRL = (4|(1412<<3)).

Once VCAP_UPDATE_CTRL.UPDATE_SHOT is cleared, then the rule has been completely written to memory and will be able to match TRI_VID keys applied to the CLM1 VCAP.

3.10.4.2 Writing an ES0 Rule

This example shows how to write an ES0 rule, consisting of ISDX entry and ES0 action, to the VCAP ES0.

The ES0 is not part of any resource sharing so it owns the entire address range 0-511. In this example, the rule is written to address 215.

Software is expected to track memory usage so that it only writes to VCAP memory which does not already contain active rules.

The ISDX entry is 30 bits wide and the action is 285 bits wide. For information about how to build a ISDX entry and ES0 action, see VCAP_ES0 Lookup, page 323.

3.10.4.2.1 Writing a Rule to ES0

Software is not allowed to modify the cache while VCAP operation is ongoing. Always check if VCAP_UPDATE_CTRL.UPDATE_SHOT is cleared before starting to modify the cache. If a previous operation is still ongoing, then wait for the UPDATE_SHOT to clear.

When a new rule is started, first clear cache's entry, action, and counter by setting VCAP_UPDATE_CTRL.CLEAR_CACHE = 1.

Write entry data to cache: VCAP_ENTRY_DAT[0][31:0] = zero-extend(entry data[29:0]).

Write entry mask to cache: VCAP_MASK_DAT[0][31:0] = zero-extend(entry mask[29:0]).

Write action to cache: VCAP_ACTION_DAT[8][31:0] = zero-extend(action[284:256]), VCAP_ACTION_DAT[7][31:0] = action[255:224], VCAP_ACTION_DAT[6][31:0] = action[223:192], VCAP_ACTION_DAT[5][31:0] = action[191:160], VCAP_ACTION_DAT[4][31:0] = action[159:128], VCAP_ACTION_DAT[3][31:0] = action[127:96], VCAP_ACTION_DAT[2][31:0] = action[95:64], VCAP_ACTION_DAT[1][31:0] = action[63:32], and VCAP_ACTION_DAT[0][31:0] = action[31:0].

Enable the rule by setting VCAP_RULE_ENA = 1.

Write cache to VCAP memory by setting VCAP_UPDATE_CTRL = (4|(215<<3)).

After VCAP_UPDATE_CTRL.UPDATE_SHOT is cleared, the rule is completely written to memory and will be able to match ISDX keys applied to the VCAP ES0.

3.11 Pipeline Points

This section describes the use of pipeline points in the processing flow through the switch core. A pipeline point defines a specific position in the processing flow where frames can be injected, extracted, discarded, or looped. These actions are special, because they define either a starting point (inject) or an ending point (extract, discard, loop) in the processing flow. A starting point identifies the point in the processing flow where the processing of the frame begins. Similarly, an ending point identifies the point in the processing flow where the processing of the frame ends.

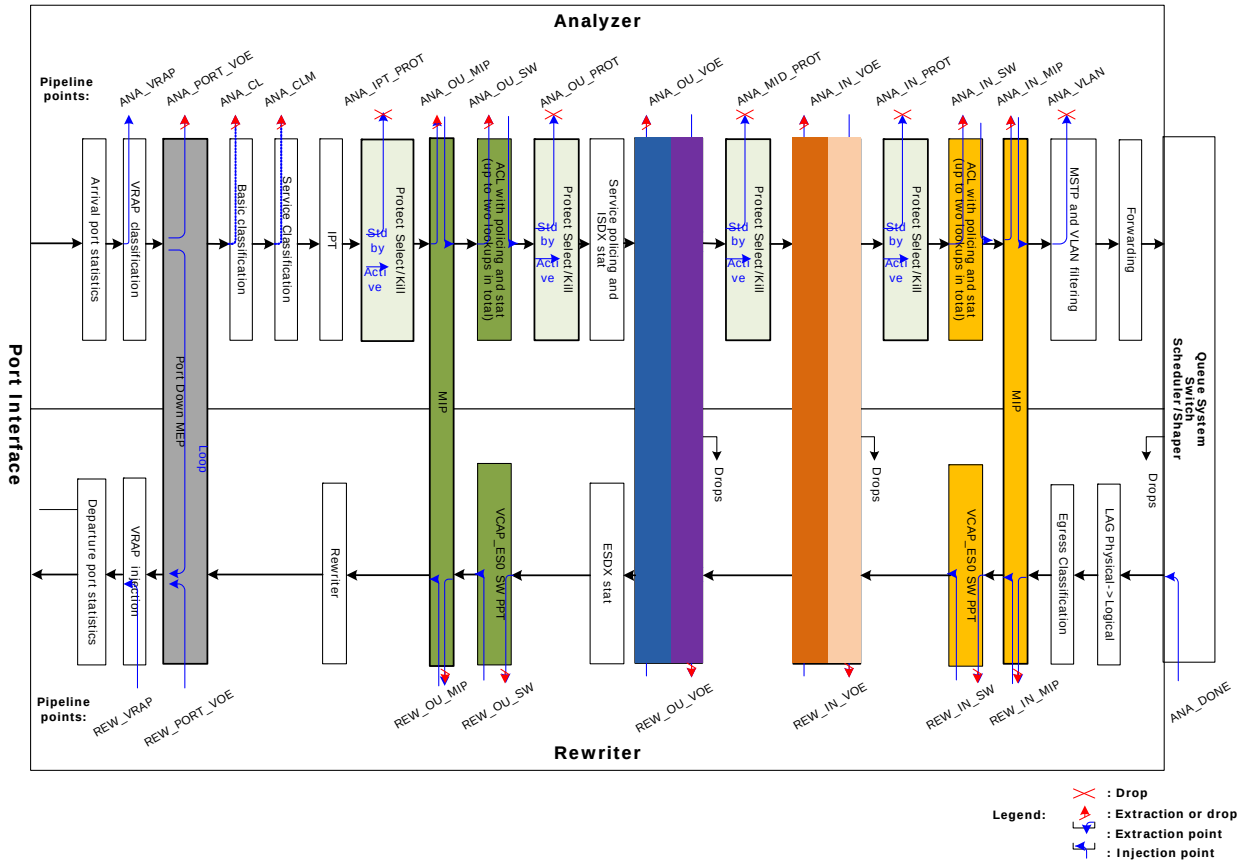
Frames subject to one of the actions injection, extraction, discarding, or looping, are assigned the associated pipeline point as well as a pipeline action defining what caused the pipeline point to be triggered. All frames are physically passed through all blocks of the processing flow, but each frame's pipeline point control determines whether it is actively processed in a given block or passed transparently to the next block; this includes which counters and policers are active.

The logical processing flow for a frame is shown in the following illustration. As an example of how the pipeline points affect the processing flow, consider a frame being discarded by the VLAN acceptance

filter in the basic classification. The frame is assigned the ANA_CL pipeline point, and it is therefore not service classified and hence not service policed nor counted in the service statistics.

The processing flow illustrates the different functional blocks and their pipeline points.

Figure 29 • **Processing Flow**



3.11.1 Pipeline Definitions

The IFH fields IFH.MISC.PIPELINE_PT and IFH.MISC.PIPELINE_ACT carry the frame's pipeline information (point and action). By default, no pipeline points are triggered and both fields are set to 0. This implies that there are no restrictions on the processing of the frame.

The following table defines the available pipeline points in the analyzer and the rewriter.

Table 41 • **Pipeline Definitions**

Pipeline Point	Block/Mechanism that uses Pipeline Point	Application
ANA_VRAP	VRAP	VRAP extraction point.
ANA_PORT_VOE	Port VOE	Port VOE extraction/discard point.
ANA_CL	Basic classifier	Filter discard point/CPU injection point with software configured IFH.
ANA_CLM	VCAP CLM/service classification	CPU injection point with software configured IFH including service classification. Extraction point related to services.
ANA_IPT_PROT	IPT protection point	Protection discard point for e.g. LAG protection (where protection discard is done before Down MEPs) controlled by port VOE.

Table 41 • **Pipeline Definitions *(continued)***

Pipeline Point	Block/Mechanism that uses Pipeline Point	Application
ANA_OU_MIP	Outer OAM half-MIP located in analyzer	Extraction/copy/injection point for analyzer half-MIP for: • UNI-N: Subscriber MIP • E-NNI: EVC MIP
ANA_OU_SW	Outer software MEP with extraction controlled by VCAP IS2	SW MEP located between the VOEs and the port for: • Extraction point for SW Down-MEP • Injection point for SW Up-MEP
ANA_OU_PROT	Outer protection point controlled by IPT	Protection discard point controlled by IPT for NNI protection
ANA_OU_VOE	Outer OAM VOE located in analyzer/VOP	Outer VOE pipeline point for: • UNI: EVC OAM injection • E-NNI: OVC OAM Injection • NNI: Path OAM Extraction
ANA_MID_PROT	Middle protection point located between outer and inner VOE controlled by IPT	Protection discard point for path protection controlled by path MEP.
ANA_IN_VOE	Inner OAM VOE located in analyzer/VOP	Inner VOE pipeline point for: • UNI: OVC OAM injection • NNI: EVC/OVC OAM segment extraction
ANA_IN_PROT	Inner protection point located after inner VOE controlled by IPT	Protection discard point. Frames are discarded after the VOEs used for discarding which exit an inactive EVC Segment
ANA_IN_SW	Inner software MEP with extraction controlled by VCAP IS2	SW MEP located between the VOEs and shared queue system for: • Extraction point for SW Down-MEP • Injection point for SW Up-MEP
ANA_IN_MIP	Inner OAM half-MIP located in analyzer	Extraction/copy/injection point for analyzer half MIP for NNI: EVC/OVC MIP.
ANA_VLAN	Pipeline point associated with VLAN handling and forwarding	Pipeline point for VLAN discard.
ANA_DONE	Pipeline point after analyzer	Injection point where analyzer is bypassed. Used for injection directly into rewriter.
REW_IN_MIP	Inner OAM half-MIP located in rewriter	Extraction/copy/injection point for rewriter half MIP function for NNI: EVC/OVC MIP.
REW_IN_SW	Inner software MEP located in rewriter controlled by VCAP_ES0	SW MEP located between the shared queue system and the VOEs for: • Injection point for SW Down-MEP • Extraction point for SW Up-MEP
REW_IN_VOE	Inner OAM VOE located in rewriter/VOP	Inner VOE pipeline point for: • UNI: OVC OAM extraction • NNI: EVC/OVC OAM segment injection
REW_OU_VOE	Outer OAM VOE in rewriter/VOP	Outer VOE pipeline point for: • UNI: EVC OAM extraction • E-NNI: OVC OAM extraction • NNI: Path OAM injection
REW_OU_SW	Outer software MEP located in rewriter controlled by VCAP_ES0	SW MEP located between the VOEs and the port for: • Injection point for SW Down-MEP • Extraction point for SW Up-MEP

Table 41 • **Pipeline Definitions *(continued)***

Pipeline Point	Block/Mechanism that uses Pipeline Point	Application
REW_OU_MIP	Outer OAM half-MIP located in rewriter	Extraction/copy/injection point for rewriter half MIP function for: • UNI-N: Subscriber MIP • E-NNI: EVC MIP
REW_SAT	Service activation testing	SAT loop point.
REW_PORT_VOE	Port VOE in rewriter VOP	Port VOE injection point.
REW_VRAP	VRAP injection point	VRAP injection point.

The following table defines the pipeline actions associated with the pipeline point.

Table 42 • **Pipeline Actions**

Pipeline Action	Description
INJ	Set when CPU-injecting a frame into the processing flow at the associated pipeline point. The processing starts at the associated pipeline point.
INJ_MASQ	Set when CPU-injecting a frame masqueraded into the processing flow at the associated pipeline point. The processing starts at the associated pipeline point.
XTR	Assigned to frames being redirected to the CPU. The processing ends at the associated pipeline point. Frames copied to the CPU are not assigned a pipeline point as the processing continues.
XTR_UPMEP	Assigned to frames being extracted to the CPU by an Up MEP (VOE). The processing ends at the associated pipeline point. Frames copied to the CPU are not assigned a pipeline point as the processing continues.
XTR_LATE_REW	Assigned to MPLS OAM frames by VCAP CLM to instruct the rewriter to terminate the processing of the frame. The processing ends at the associated pipeline point.
LBK_ASM	Assigned by Up MEPs (VOE) requiring the frame to be looped. This action implies that the processing in the rewriter is ended at the associated pipeline point and that the processing in the analyzer after looping the frame starts at the corresponding analyzer pipeline point.
LBK_QS	Assigned by Down MEPs (VOE) requiring the frame to be looped. This action implies that the processing in the analyzer is ended at the associated pipeline point and that the processing in the rewriter after looping the frame starts at the corresponding rewriter pipeline point.

Note: A frame cannot be assigned both a starting point and an ending point at the same side of the queue system. For instance, a frame cannot be injected in the analyzer at ANA_CLM and then extracted at ANA_IN_SW. However, a frame can be injected in the analyzer and then extracted in the rewriter.

3.12 Analyzer

The following sections provides information about the functional aspects of the analyzer (ANA). The analyzer evokes different actions based on the contents of the frame fields. The analyzer is organized into the following blocks.

- VCAP CLM: VCAP Classification matching—keys and actions
- ANA_CL: Analyzer classifier
- ANA_L3: VLAN and MSTP
- VCAP LPM: VCAP longest prefix matching—keys and actions
- ANA_L3: IP routing
- VCAP IS2: VCAP Ingress Stage 2—keys and actions
- ANA_ACL: Access Control Lists
- ANA_L2: Analyzer Layer 2 forwarding and learning
- ANA_AC: Analyzer Actions—forwarding, policing, statistics

3.12.1 Initializing the Analyzer

Before the analyzer can be configured, the RAM-based configuration registers must be initialized by setting ANA_AC:RAM_CTRL:RAM_INIT.RAM_ENA and ANA_AC:RAM_CTRL:RAM_INIT.RAM_INIT to 1.

The ANA_AC:RAM_CTRL:RAM_INIT.RAM_INIT register is reset by hardware when initialization is complete.

For information about initializing VCAP CLM, VCAP IS2, and VCAP LPM, see Bringing Up VCAP, page 57.

3.13 VCAP CLM Keys and Actions

The VCAP CLM is part of the analyzer and enables frame classification using VCAP functionality. This section provides an overview of all available VCAP CLM keys, followed by information about each key and action.

For information about how to select which key to use and how the VCAP CLM interacts with other parts of the analyzer classifier, see VCAP CLM Processing, page 115.

VCAP CLM supports a number of different keys that can be used for different purposes. The keys are of type X1, X2, X4, X8, or X16, depending on the number of words each key uses. The keys are grouped after size as shown in the following table.

Table 43 • **VCAP CLM Keys and Sizes**

Key Name	Key Size	Number of Words	Key Type
SGL_MLBS	35 bits	1 word	X1 type
TRI_VID	69 bits	2 words	X2 type
DBL_MLBS	68 bits		
TRI_VID_IDX	69 bits		
MLL	136 bits	4 words	X4 type
TRI_MLBS	129 bits		
PURE_5TUPLE_IP4	137 bits		
CUSTOM_4	136 bits		
LL_FULL	272 bits	8 words	X8 type
NORMAL	266 bits		
NORMAL_5TUPLE_IP4	255 bits		
CUSTOM_2	272 bits		
NORMAL_7TUPLE	536 bits	16 words	X16 type
CUSTOM_1	527 bits		

3.13.1 Keys Overview

The following table lists all VCAP CLM keys and fields and provides a quick overview of which fields are available in which keys.

When programming an entry in VCAP CLM, the associated key fields listed must be programmed in the listed order with the first field in the table starting at bit 0 in the entry.

Table 44 • **VCAP CLM Key Overview**

Field Name	Short Description	Size	SGL_MLBS	TRI_VID	TRI_VID_IDX	DBL_MLBS	MLL	TRI_MLBS	PURE_5TUPLE_IP4	CUSTOM_4	LL_FULL	NORMAL	NORMAL_5TUPLE_IP4	CUSTOM_2	NORMAL_7TUPLE	CUSTOM_1
X2_TYPE	X2 type (each entry uses 2 words)	2		x	x	x										
X4_TYPE	X4 type (each entry uses 4 words)	2					x	x	x	x						
X8_TYPE	X8 type (each entry uses 8 words)	2									x	x	x	x		
X16_TYPE	X16 type (each entry uses 16 words)	1													x	x
FIRST	Set for first lookup	1	x	x	x	x		x	x	x	x	x	x	x	x	x
IGR_PORT	Ingress port number	6		x			x				x					
G_IDX_IS_SERVICE	Set if G_IDX is set to ISDX	1			x				x	x		x	x	x	x	x
G_IDX	Generic index	12	x		x	x		x	x	x		x	x	x	x	x
IGR_PORT_MASK_SEL	Mode selector for IGR_PORT_MASK	2										x	x		x	
IGR_PORT_MASK	Ingress port mask	53										x	x		x	
L2_MC	Multicast DMAC address	1										x	x		x	
L2_BC	Broadcast DMAC address	1										x	x		x	
TPID0	TPID identifier from first tag	3		x	x		x				x	x	x		x	
PCP0	PCP from first tag	3		x							x	x	x		x	
DEI0	DEI from first tag	1		x							x	x	x		x	
VID0	VID from first tag	12		x	x		x				x	x	x		x	
TPID1	TPID identifier from second tag	3		x	x		x				x	x	x		x	
PCP1	PCP from second tag	3		x							x	x	x		x	
DEI1	DEI from second tag	1		x							x	x	x		x	
VID1	VID from second tag	12		x	x		x				x	x	x		x	
TPID2	Tag protocol identifier from third tag	3		x	x						x	x	x		x	
PCP2	PCP from third tag	3		x							x	x	x		x	
DEI2	DEI from third tag	1		x							x	x	x		x	
VID2	VID from third tag	12		x	x						x	x	x		x	
DST_ENTRY	Set if destination entry	1										x				
L2_DMAC	Destination MAC address	48					x				x				x	
L2_SMAC	Source MAC address	48					x				x	x			x	
ETYPE_MPLS	EtherType identifier	2					x									
ETYPE_FULL	EtherType identifier	3		x												
IP_MC	Multicast DIP address	1										x	x		x	
ETYPE_LEN	ETYPE encoded frame	1									x	x			x	

Table 44 • **VCAP CLM Key Overview *(continued)***

Field Name	Short Description	Size	SGL_MLBS	TRI_VID	TRI_VID_IDX	DBL_MLBS	MLL	TRI_MLBS	PURE_5TUPLE_IP4	CUSTOM_4	LL_FULL	NORMAL	NORMAL_5TUPLE_IP4	CUSTOM_2	NORMAL_7TUPLE	CUSTOM_1
ETYPE	EtherType, overloaded for other frame types	16									x	x			x	
IP_SNAP	IP or SNAP frame	1									x	x			x	
IP4	IPv4 frame	1									x	x	x		x	
LBL0	Label from MPLS Label Stack entry 0	20	x			x		x								
TC0	TC bits from MPLS Label Stack entry 0	3				x		x								
SBIT0	S-bit from MPLS Label Stack entry 0	1	x			x		x								
TTL0_EXPIRY	Set if TTL<=1 for MPLS Label Stack entry 0	1	x			x		x								
LBL1	Label from entry 1	20				x		x								
TC1	TC bits from entry 1	3				x		x								
SBIT1	S-bit from entry 1	1				x		x								
TTL1_EXPIRY	Set if TTL<=1 for entry 1	1				x		x								
LBL2	Label from entry 2	20						x								
TC2	TC bits from entry 2	3						x								
SBIT2	S-bit from entry 2	1						x								
TTL2_EXPIRY	Set if TTL<=1 for entry 2	1						x								
RSV_LBL_VAL	Reserved label value	4						x								
CW_ACH	CW/ACH after label with S-bit set	32						x								
RSV_LBL_POS	Reserved label position	3				x		x								
L3_FRAGMENT	Fragmented IPv4 frame	1							x		x	x	x		x	
L3_FRAG_OFS_GT0	Frame is not the first fragment	1							x		x	x	x		x	
L3_OPTIONS	IPv4 frame with options	1							x		x	x	x		x	
L3_DSCP	Frame's DSCP value	6							x		x	x	x		x	
L3_IP4_DIP	Destination IP address	32							x		x		x			
L3_IP4_SIP	SIP address, overloaded for other frame types	32							x		x	x	x			
L3_IP6_DIP	Destination IP address	128													x	
L3_IP6_SIP	Source IP address	128													x	
L3_IP_PROTO	IP protocol / next header	8							x				x			
TCP_UDP	TCP/UDP frame	1									x	x	x		x	
TCP	TCP frame	1									x	x	x		x	
L4_SPORT	TCP/UDP source port	16									x	x			x	

Table 44 • **VCAP CLM Key Overview *(continued)***

Field Name	Short Description	Size	SGL_MLBS	TRI_VID	TRI_VID_IDX	DBL_MLBS	MLL	TRI_MLBS	PURE_5TUPLE_IP4	CUSTOM_4	LL_FULL	NORMAL	NORMAL_5TUPLE_IP4	CUSTOM_2	NORMAL_7TUPLE	CUSTOM_1
L4_RNG	Range checker mask	8							x			x	x		x	
IP_PAYLOAD_5TUPLE	Payload bytes after IP header	32							x				x			
OAM_Y1731	Set if frame's EtherType = 0x8902	1			x											
OAM_MEL_FLAGS	Encoding of MD level/MEG level (MEL)	7			x											
CUSTOM1	64 bytes payload	512														x
CUSTOM2	32 bytes payload	256												x		
CUSTOM4	15 bytes payload	120								x						

3.13.2 VCAP CLM X1 Key Details

The SGL_MLBS key is the only X1 key, so it does not have a type field.

The following table lists details about the fields applicable to the SGL_MLBS key.

Table 45 • **VCAP CLM X1 Key Details**

Field Name	Description	Size	SGL_MLBS
FIRST	Selects between entries relevant for first and second lookup. Set for first lookup, cleared for second lookup.	1	x
G_IDX	Generic index used to bind together entries across VCAP CLM lookups. G_IDX is calculated based on fields in VCAP CLM action from previous VCAP CLM lookup: NXT_IDX_CTRL. NXT_IDX. Default value is configurable in ANA_CL::CLM_MISC_CTRL.CLM_GIDX_DEF_SEL. Can be zero, logical port number, or masqueraded port number.	12	x
LBL0	MPLS Label Stack entry 0 - Label (Top Label)	20	x
SBIT0	MPLS Label Stack entry 0 - S-bit (Top Label)	1	x
TTL0_EXPIRY	Set if TTL<=1 for MPLS Label Stack entry 0	1	x

3.13.3 VCAP CLM X2 Key Details

The X2 keys include an X2_TYPE field, which is used to tell the difference between the keys. It takes a unique value for each key. The following table lists details about the fields applicable to these keys.

Table 46 • **VCAP CLM X2 Key Details**

Field Name	Description	Size	TRI_VID	DBL_MLBS	TRI_VID_IDX
X2_TYPE	X2 type: 0: TRI_VID 1: DBL_MLBS 2: Reserved 3: TRI_VID_IDX	2	x	x	x
FIRST	Selects between entries relevant for first and second lookup. Set for first lookup, cleared for second lookup.	1	x	x	x
IGR_PORT	Logical ingress port number retrieved from ANA_CL::PORT_ID_CFG.LPORT_NUM.	6	x		
G_IDX	Generic index used to bind together entries across VCAP CLM lookups. G_IDX is calculated based on fields in VCAP CLM action from previous VCAP CLM lookup: NXT_IDX_CTRL. NXT_IDX. Default value is configurable in ANA_CL::CLM_MISC_CTRL.CLM_GIDX_DEF_SEL. Can be zero, logical port number, or masqueraded port number.	12		x	x
TPID0	Tag protocol identifier of the frame's first tag (outer tag): 0: Untagged. 1: 0x8100. 4: 0x88A8. 5: Custom value 1. 6: Custom value 2. 7: Custom value 3.	3	x		x
PCP0	By default PCP from frame but it is selectable per lookup in VCAP CLM whether to use the current classified PCP instead (ANA_CL::ADV_CL_CFG.USE_CL_TCI0_ENA).	3	x		
DEI0	By default DEI from frame but it is selectable per lookup in VCAP CLM whether to use the current classified DEI instead (ANA_CL::ADV_CL_CFG.USE_CL_TCI0_ENA).	1	x		
VID0	By default VID from frame but it is selectable per lookup in VCAP CLM whether to use the current classified VID instead (ANA_CL::ADV_CL_CFG.USE_CL_TCI0_ENA). For untagged frames, VID0 is set to 0.	12	x		x
TPID1	Tag protocol identifier of the frame's second tag (tag after outer tag): 0: No second tag. 1: 0x8100. 4: 0x88A8. 5: Custom value 1. 6: Custom value 2. 7: Custom value 3.	3	x		x

Table 46 • **VCAP CLM X2 Key Details *(continued)***

Field Name	Description	Size	TRI_VID	DBL_MLBS	TRI_VID_IDX
PCP1	Frame's PCP from second tag.	3	x		
DEI1	Frame's DEI from second tag.	1	x		
VID1	Frame's VID from second tag.	12	x		
TPID2	Tag protocol identifier of the frame's third tag: 0: No third tag. 1: 0x8100. 4: 0x88A8. 5: Custom value 1. 6: Custom value 2. 7: Custom value 3. Overloading for TRI_VID: For OAM Y.1731 frames (ETYPE_FULL = 5) without a third tag (TPID2 = 0), VID2[6:0] is set to OAM MEL flags: Encoding of MD level/MEG level (MEL). One bit for each level where lowest level encoded as zero. The following keys can be generated: MEL = 0: 0b0000000. MEL = 1: 0b0000001. MEL = 2: 0b0000011. MEL = 3: 0b0000111. MEL = 4: 0b0001111. MEL = 5: 0b0011111. MEL = 6: 0b0111111. MEL = 7: 0b1111111. Together with the mask, the following types of rules can be created: Exact match. Fx. MEL = 2: 0b0000011. Below. Fx. MEL< = 4: 0b000XXXX. Above. Fx. MEL> = 5: 0bXX11111. Between. Fx. 3< = MEL< = 5: 0b00XX111, where "X" means don't care.	3	x		x
PCP2	Frame's PCP from third tag.	3	x		
DEI2	Frame's DEI from third tag.	1	x		
VID2	Frame's VID from third tag.	12	x		
ETYPE_FULL	EtherType identifier: 0: IPv4frame (EtherType = 0x0800). 1: IPv6 frame (EtherType = 0x86DD). 2: Downstream assigned label (EtherType = 0x8847). 3: Upstream assigned label (EtherType = 0x8848). 4: LCC/SNAP (EtherType < 0x0600). 5: OAM Y.1731 (EtherType = 0x8902). 6: (R)ARP (EtherType = 0x0806 or 0x8035). 7: Other.	3	x		
LBL0	MPLS Label Stack entry 0 - Label (Top Label).	20		x	
TC0	MPLS Label Stack entry 0 - TC bits (Top Label).	3		x	
SBIT0	MPLS Label Stack entry 0 - S-bit (Top Label).	1		x	
TTL0_EXPIRY	Set if TTL<=1 for MPLS Label Stack entry 0.	1		x	

Table 46 • **VCAP CLM X2 Key Details *(continued)***

Field Name	Description	Size	TRI_VID	DBL_MLBS	TRI_VID_IDX
LBL1	MPLS Label Stack entry 1 - Label.	20		x	
TC1	MPLS Label Stack entry 1 - TC bits.	3		x	
SBIT1	MPLS Label Stack entry 1 - S-bit.	1		x	
TTL1_EXPIRY	Set if TTL<=1 for MPLS Label Stack entry 1.	1		x	
RSV_LBL_POS	Reserved label position: 0: Reserved label at position 0 seen or skipped. 1: Reserved label at position 1 seen or skipped. 2: Reserved label at position 2 seen or skipped. 3: Reserved label at position 3 seen. 4: Reserved. 5: No reserved label seen. Label at top of stack is position 0, followed by 1, 2, and 3. In order for a reserved label to be skipped, either ANA_CL::MPLS_RSV_LBL_CFG[<label>].RSVD_LBL_SKIP_ENA or ANA_CL::MPLS_MISC_CFG.CLM_RSVD_LBL_SKIP_ENA[<clm idx>] must be set.	3		x	
OAM_Y1731	Set if frame's EtherType = 0x8902.	1			x
OAM_MEL_FLAGS	Encoding of MD level/MEG level (MEL). One bit for each level where lowest level encoded as zero. The following keys can be generated: MEL=0: 0x0000000 MEL=1: 0x0000001 MEL=2: 0x0000011 MEL=3: 0x0000111 MEL=4: 0x0001111 MEL=5: 0x0011111 MEL=6: 0x0111111 MEL=7: 0x1111111 Together with the mask, the following kinds of rules may be created: - Exact match. Fx. MEL=2: 0x0000011 - Below. Fx. MEL<=4: 0x000XXXX - Above. Fx. MEL>=5: 0xXX11111 - Between. Fx. 3<= MEL<=5: 0x00XX111, where 'X' means don't care. Overloading: For non-Y1731 OAM frames (OAM_Y1731 = 0), OAM_MEL_FLAGS encodes an EtherType identifier: 0: IPv4frame (EtherType = 0x0800) 1: IPv6 frame (EtherType = 0x86DD) 2: Downstream assigned label (EtherType = 0x8847) 3: Upstream assigned label (EtherType = 0x8848) 4: LCC/SNAP (EtherType < 0x0600) 5: (R)ARP (EtherType = 0x0806 or 0x8035) 6: Reserved 7: Other	7			x

3.13.4 VCAP CLM X4 Key Details

The X4 keys include an X4_TYPE field, which is used to tell difference between the keys. It takes a unique value for each key. The following table lists details about the fields applicable to these keys.

Table 47 • **VCAP CLM X4 Key Details**

Field Name	Description	Size	MLL	TRI_MLBS	PURE_5TUPLE_IP4	CUSTOM_4
X4_TYPE	X4 type: 0: MLL. 1: TRI_MLBS. 2: PURE_5TUPLE_IP4. 3: CUSTOM_4.	2	x	x	x	x
FIRST	Selects between entries relevant for first and second lookup. Set for first lookup, cleared for second lookup.	1		x	x	x
IGR_PORT	Logical ingress port number retrieved from ANA_CL::PORT_ID_CFG.LPORT_NUM.	6	x			
G_IDX_IS_SERVICE	Set if the VCAP CLM action from the previous VCAP CLM lookup specified ISDX to be used as G_IDX.	1			x	x
G_IDX	Generic index used to bind together entries across VCAP CLM lookups. G_IDX is calculated based on fields in VCAP CLM action from previous VCAP CLM lookup: NXT_IDX_CTRL. NXT_IDX. Default value is configurable in ANA_CL::CLM_MISC_CTRL.CLM_GIDX_DEF_SEL. Can be zero, logical port number, or masqueraded port number.	12		x	x	x
TPID0	Tag protocol identifier of the frame's first tag (outer tag): 0: Untagged. 1: 0x8100. 4: 0x88A8. 5: Custom value 1. 6: Custom value 2. 7: Custom value 3.	3	x			
VID0	By default VID from frame but it is selectable per lookup in VCAP CLM whether to use the current classified VID instead (ANA_CL::ADV_CL_CFG.USE_CL_TCI0_ENA). For untagged frames, VID0 is set to 0.	12	x			
TPID1	Tag protocol identifier of the frame's second tag (tag after outer tag): 0: No second tag. 1: 0x8100. 4: 0x88A8. 5: Custom value 1. 6: Custom value 2. 7: Custom value 3.	3	x			
VID1	Frame's VID from second tag.	12	x			
L2_DMAC	Destination MAC address.	48	x			

Table 47 • **VCAP CLM X4 Key Details *(continued)***

Field Name	Description	Size	MLL	TRI_MLBS	PURE_5TUPLE_IP4	CUSTOM_4
L2_SMAC	Source MAC address.	48	x			
ETYPE_MPLS	EtherType identifier: 0: Non-MPLS. 1: Downstream Assigned Label (EtherType = 0x8847). 2: Upstream Assigned Label (EtherType = 0x8848).	2	x			
LBL0	MPLS Label Stack entry 0 - Label (Top Label).	20		x		
TC0	MPLS Label Stack entry 0 - TC bits (Top Label).	3		x		
SBIT0	MPLS Label Stack entry 0 - S-bit (Top Label).	1		x		
TTL0_EXPIRY	Set if TTL<=1 for MPLS Label Stack entry 0.	1		x		
LBL1	MPLS Label Stack entry 1 - Label.	20		x		
TC1	MPLS Label Stack entry 1 - TC bits.	3		x		
SBIT1	MPLS Label Stack entry 1 - S-bit.	1		x		
TTL1_EXPIRY	Set if TTL<=1 for MPLS Label Stack entry 1.	1		x		
LBL2	MPLS Label Stack entry 2 - Label.	20		x		
TC2	MPLS Label Stack entry 2 - TC bits.	3		x		
SBIT2	MPLS Label Stack entry 2 - S-bit.	1		x		
TTL2_EXPIRY	Set if TTL<=1 for MPLS Label Stack entry 2.	1		x		
RSV_LBL_VAL	Reserved label value. Only valid if RSV_LBL_POS > 0.	4		x		
CW_ACH	The 32 bits following the label with S-bit set.	32		x		
RSV_LBL_POS	Reserved label position: 0: Reserved label at position 0 seen or skipped. 1: Reserved label at position 1 seen or skipped. 2: Reserved label at position 2 seen or skipped. 3: Reserved label at position 3 seen. 4: Reserved. 5: No reserved label seen. Label at top of stack is position 0, followed by 1, 2, and 3. In order for a reserved label to be skipped, either ANA_CL::MPLS_RSV_LBL_CFG[<label>].RSVD_LBL_SKIP_ENA or ANA_CL::MPLS_MISC_CFG.CLM_RSVD_LBL_SKIP_ENA[<clm idx>] must be set.	3		x		
L3_FRAGMENT	Set if IPv4 frame is fragmented (More Fragments flag = 1 or Fragments Offset > 0).	1			x	
L3_FRAG_OFS_GT0	Set if IPv4 frame is fragmented but not the first fragment (Fragments Offset > 0)	1			x	
L3_OPTIONS	Set if IPv4 frame contains options (IP len > 5). IP options are not parsed.	1			x	

Table 47 • **VCAP CLM X4 Key Details *(continued)***

Field Name	Description	Size	MLL	TRI_MLBS	PURE_5TUPLE_IP4	CUSTOM_4
L3_DSCP	By default DSCP from frame. Per match, selectable to be current classified DSCP (ANA_CL::ADV_CL_CFG.USE_CL_DSCP_ENA).	6			x	
L3_IP4_DIP	IPv4 frames: Destination IPv4 address. IPv6 frames: Destination IPv6 address, bits 31:0.	32			x	
L3_IP4_SIP	Overloaded field for different frame types: LLC frames (ETYPE_LEN = 0 and IP_SNAP = 0): L3_IP4_SIP = [CTRL, PAYLOAD[0:2]] SNAP frames (ETYPE_LEN = 0 and IP_SNAP=1): L3_IP4_SIP = [PID[2:0], PAYLOAD[0]] IPv4 frames (ETYPE_LEN=1, IP_SNAP=1, and IP4=1): L3_IP4_SIP = source IPv4 address IPv6 frames (ETYPE_LEN=1, IP_SNAP=1, IP4=0): L3_IP4_SIP = source IPv6 address, bits 31:0 Other frames (ETYPE_LEN=1, IP_SNAP=0): L3_IP4_SIP = PAYLOAD[0:3].	32			x	
L3_IP_PROTO	IPv4 frames (IP4=1): IP protocol. IPv6 frames (IP4=0): Next header.	8			x	
L4_RNG	Range mask. Range types: SPORT, DPORT, SPORT or DPORT, VID, DSCP, custom. Input into range checkers is taken from frame except DSCP value which is the remapped DSCP value from basic classification.	8			x	
IP_PAYLOAD_5TUPLE	Payload bytes after IP header. IPv4 options are not parsed so payload is always taken 20 bytes after the start of the IPv4 header.	32			x	
CUSTOM4	15 bytes payload starting from current frame pointer position, as controlled by VCAP CLM actions. Note that if frame_type==ETH, then payload is retrieved from a position following the Ethernet layer, for example, after DMAC, SMAC, 0-3 VLAN tags and EtherType.	120				x

3.13.5 VCAP CLM X8 Key Details

The X8 keys include an X8_TYPE field, which is used to tell difference between the keys. It takes a unique value for each key. The following table lists details about the fields applicable to these keys.

Table 48 • **VCAP CLM X8 Key Details**

Field Name	Description	Size	LL_FULL	NORMAL	NORMAL_5TUPLE_IP4	CUSTOM_2
X8_TYPE	X8 type: 0: LL_FULL. 1: NORMAL. 2: NORMAL_5TUPLE_IP4. 3: CUSTOM_2.	2	x	x	x	x
FIRST	Selects between entries relevant for first and second lookup. Set for first lookup, cleared for second lookup.	1	x	x	x	x
IGR_PORT	Logical ingress port number retrieved from ANA_CL::PORT_ID_CFG.LPORT_NUM.	6	x			
G_IDX_IS_SERVICE	Set if the VCAP CLM action from the previous VCAP CLM lookup specified ISDX to be used as G_IDX.	1		x	x	x
G_IDX	Generic index used to bind together entries across VCAP CLM lookups. G_IDX is calculated based on fields in VCAP CLM action from previous VCAP CLM lookup: NXT_IDX_CTRL. NXT_IDX. Default value is configurable in ANA_CL::CLM_MISC_CTRL.CLM_GIDX_DEF_SEL. Can be zero, logical port number, or masqueraded port number.	12		x	x	x

Table 48 • **VCAP CLM X8 Key Details *(continued)***

Field Name	Description	Size	LL_FULL	NORMAL	NORMAL_5TUPLE_IP4	CUSTOM_2
IGR_PORT_MASK_SEL	Mode selector for IGR_PORT_MASK. 0: Default setting. 1: Set for frames received from a loopback device, i.e. LBK_DEV*. 2: Set for masqueraded frames if ANA_CL::CLM_MISC_CTRL.MASQ_IGR_MASK_ENA == 1. A masqueraded frame is identified by the following criteria: (ifh.fwd.dst_mode == inject && ifh.src_port != physical_src_port) || (misc.pipeline_act == inj_masq) 3: Set for the following frame types: 3.a: CPU injected frames and ANA_CL::CLM_MISC_CTRL.CPU_IGR_MASK_ENA == 1. 3.b: Frames received from VD0 or VD1 and ANA_CL::CLM_MISC_CTRL.VD_IGR_MASK_ENA == 1. 3.c: Frame received on a loopback device and ANA_CL::CLM_MISC_CTRL.LBK_IGR_MASK_SEL3_ENA == 1. If a frame fulfills multiple of above criteria, then higher value of IGR_PORT_MASK_SEL takes precedence.	2		x	x	
IGR_PORT_MASK	Ingress port mask. IGR_PORT_MASK_SEL == 0: Each bit in the mask correspond to physical ingress port. IGR_PORT_MASK_SEL == 1: Each bit in the mask correspond to the physical port which the frame was looped on. IGR_PORT_MASK_SEL == 2: Each bit in the mask correspond to the masqueraded port. If IGR_PORT_MASK_SEL == 3: Bit 0: Physical src port == CPU0. Bit 1: Physical src port == CPU1. Bit 2: Physical src port == VD0. Bit 3: Physical src port == VD1. Bits 4:9: Src port (possibly masqueraded). Bits 44:48: ifh.misc.pipeline_pt. Bits 49:51: ifh.misc.pipeline_act. Bits 52: Reserved.	53		x	x	
L2_MC	Set if frame's destination MAC address is a multicast address (bit 40 = 1).	1		x	x	
L2_BC	Set if frame's destination MAC address is the broadcast address (FF-FF-FF-FF-FF-FF).	1		x	x	

Table 48 • **VCAP CLM X8 Key Details *(continued)***

Field Name	Description	Size	LL_FULL	NORMAL	NORMAL_5TUPLE_IP4	CUSTOM_2
TPID0	Tag protocol identifier of the frame's first tag (outer tag): 0: Untagged. 1: 0x8100. 4: 0x88A8. 5: Custom value 1. 6: Custom value 2. 7: Custom value 3.	3	x	x	x	
PCP0	By default PCP from frame but it is selectable per lookup in VCAP CLM whether to use the current classified PCP instead (ANA_CL::ADV_CL_CFG.USE_CL_TCI0_ENA).	3	x	x	x	
DEI0	By default DEI from frame but it is selectable per lookup in VCAP CLM whether to use the current classified DEI instead (ANA_CL::ADV_CL_CFG.USE_CL_TCI0_ENA).	1	x	x	x	
VID0	By default VID from frame but it is selectable per lookup in VCAP CLM whether to use the current classified VID instead (ANA_CL::ADV_CL_CFG.USE_CL_TCI0_ENA). For untagged frames, VID0 is set to 0.	12	x	x	x	
TPID1	Tag protocol identifier of the frame's second tag (tag after outer tag): 0: No second tag. 1: 0x8100. 4: 0x88A8. 5: Custom value 1. 6: Custom value 2. 7: Custom value 3.	3	x	x	x	
PCP1	Frame's PCP from second tag.	3	x	x	x	
DEI1	Frame's DEI from second tag.	1	x	x	x	
VID1	Frame's VID from second tag.	12	x	x	x	
TPID2	Tag protocol identifier of the frame's third tag: 0: No third tag. 1: 0x8100. 4: 0x88A8. 5: Custom value 1. 6: Custom value 2. 7: Custom value 3.	3	x	x	x	
PCP2	Frame's PCP from third tag.	3	x	x	x	
DEI2	Frame's DEI from third tag.	1	x	x	x	
VID2	Frame's VID from third tag.	12	x	x	x	

Table 48 • **VCAP CLM X8 Key Details *(continued)***

Field Name	Description	Size	LL_FULL	NORMAL	NORMAL_5TUPLE_IP4	CUSTOM_2
DST_ENTRY	Selects whether the frame's destination or source information is used for fields L2_SMAC and L3_IP4_SIP. If set, L2_SMAC contains the frame's destination MAC address and L3_IP4_SIP contains the frame's destination IP address. The setting is controlled by ANA_CL::ADV_CL_CFG and VCPA_CLM_action.NXT_KEY_TYPE.	1		x		
L2_DMAC	Destination MAC address.	48	x			
L2_SMAC	Source MAC address. Note that - optionally - L2_SMAC may contain the destination MAC address, see DST_ENTRY.	48	x	x		
IP_MC	IPv4 frames: Set if frame's destination IP address is an IPv4 multicast address (0xE /4). IPv6 frames: Set if frame's destination IP address is an IPv6 multicast address (0xFF /8).	1		x	x	
ETYPE_LEN	Frame type flag indicating that the frame is EtherType encoded. Set if frame has EtherType >= 0x600.	1	x	x		
ETYPE	Overloaded field for different frame types: LLC frames (ETYPE_LEN=0 and IP_SNAP=0): ETYPE = [DSAP, SSAP] SNAP frames (ETYPE_LEN=0 and IP_SNAP=1): ETYPE = PID[4:3] from SNAP header TCP/UDP IPv4 or IPv6 frames (ETYPE_LEN=1, IP_SNAP=1, and TCP_UDP=1): ETYPE = TCP/UDP destination port Non-TCP/UDP IPv4 or IPv6 frames (ETYPE_LEN=1, IP_SNAP=1, and TCP_UDP=0): ETYPE = IP protocol Other frames (ETYPE_LEN=1 and IP_SNAP=0): ETYPE = Frame's EtherType.	16	x	x		
IP_SNAP	Frame type flag indicating that the frame is either an IP frame or a SNAP frame. IP frames (ETYPE_LEN=1): Set if (EtherType= 0x0800 and IP version = 4) or (ETYPE = 0x86DD and IP version = 6) SNAP frames (ETYPE_LEN=0): Set if LLC header contains AA-AA-03.	1	x	x		
IP4	Frame type flag indicating the frame is an IPv4 frame. Set if frame is IPv4 frame (EtherType = 0x800 and IP version = 4).	1	x	x	x	

Table 48 • **VCAP CLM X8 Key Details *(continued)***

Field Name	Description	Size	LL_FULL	NORMAL	NORMAL_5TUPLE_IP4	CUSTOM_2
L3_FRAGMENT	Set if IPv4 frame is fragmented (More Fragments flag = 1 or Fragments Offset > 0).	1	x	x	x	
L3_FRAG_OFS_GT0	Set if IPv4 frame is fragmented but not the first fragment (Fragments Offset > 0).	1	x	x	x	
L3_OPTIONS	Set if IPv4 frame contains options (IP len > 5). IP options are not parsed.	1	x	x	x	
L3_DSCP	By default DSCP from frame. Per match, selectable to be current classified DSCP (ANA_CL::ADV_CL_CFG.USE_CL_DSCP_ENA).	6	x	x	x	
L3_IP4_DIP	IPv4 frames: Destination IPv4 address. IPv6 frames: Destination IPv6 address, bits 31:0.	32	x		x	
L3_IP4_SIP	Overloaded field for different frame types: LLC frames (ETYPE_LEN=0 and IP_SNAP=0): L3_IP4_SIP = [CTRL, PAYLOAD[0:2]] SNAP frames (ETYPE_LEN=0 and IP_SNAP=1): L3_IP4_SIP = [PID[2:0], PAYLOAD[0]] IPv4 frames (ETYPE_LEN=1, IP_SNAP=1, and IP4=1): L3_IP4_SIP = source IPv4 address IPv6 frames (ETYPE_LEN=1, IP_SNAP=1, IP4=0): L3_IP4_SIP = source IPv6 address, bits 31:0 Other frames (ETYPE_LEN=1, IP_SNAP=0): L3_IP4_SIP = PAYLOAD[0:3] Note that, optionally, L3_IP4_SIP may contain the destination IP address for IP frames. See DST_ENTRY.	32	x	x	x	
L3_IP_PROTO	IPv4 frames (IP4=1): IP protocol. IPv6 frames (IP4=0): Next header.	8			x	
TCP_UDP	Frame type flag indicating the frame is a TCP or UDP frame. Set if frame is IPv4/IPv6 TCP or UDP frame (IP protocol/next header equals 6 or 17). Overloading: Set to 1 for OAM Y.1731 frames.	1	x	x	x	
TCP	Frame type flag indicating the frame is a TCP frame. Set if frame is IPv4 TCP frame (IP protocol = 6) or IPv6 TCP frames (Next header = 6).	1	x	x	x	

Table 48 • **VCAP CLM X8 Key Details *(continued)***

Field Name	Description	Size	LL_FULL	NORMAL	NORMAL_5TUPLE_IP4	CUSTOM_2
L4_SPORT	TCP/UDP source port. Overloading: OAM Y.1731 frames: L4_SPORT = OAM MEL flags, see below. TCP_UDP is at the same time set to 1 to indicate the overloading. OAM MEL flags: Encoding of MD level/MEG level (MEL). One bit for each level where lowest level encoded as zero. The following keys can be generated: MEL=0: 0b0000000. MEL=1: 0b0000001. MEL=2: 0b0000011. MEL=3: 0b0000111. MEL=4: 0b0001111. MEL=5: 0b0011111. MEL=6: 0b0111111. MEL=7: 0b1111111. Together with the mask, the following kinds of rules may be created: Exact match. Fx. MEL=2: 0b0000011. Below. Fx. MEL<=4: 0b000XXXX. Above. Fx. MEL>=5: 0bXX11111. Between. Fx. 3<= MEL<=5: 0b00XX111, where 'X' means don't care.	16	x	x		
L4_RNG	Range mask. Range types: SPORT, DPORT, SPORT or DPORT, VID, DSCP, custom. Input into range checkers is taken from frame except DSCP value which is the remapped DSCP value from basic classification.	8		x	x	
IP_PAYLOAD_5TUPLE	Payload bytes after IP header. IPv4 options are not parsed so payload is always taken 20 bytes after the start of the IPv4 header.	32			x	
CUSTOM2	32 bytes payload starting from current frame pointer position, as controlled by VCAP CLM actions. Note that if frame_type==ETH, then payload is retrieved from a position following the Ethernet layer; for example, after DMAC, SMAC, 0-3 VLAN tags and EtherType.	256				x

3.13.6 VCAP CLM X16 Key Details

The X16 keys include an X16_TYPE field, which is used to tell difference between the keys. It takes a unique value for each key. The following table lists details about the fields applicable to these keys.

Table 49 • **VCAP CLM X16 Key Details**

Field Name	Description	Size	NORMAL_7TUPLE	CUSTOM_1
X16_TYPE	X16 type. 0: NORMAL_7TUPLE. 1: CUSTOM_1.	1	x	x
FIRST	Selects between entries relevant for first and second lookup. Set for first lookup, cleared for second lookup.	1	x	x
G_IDX_IS_SERVICE	Set if the VCAP CLM action from the previous VCAP CLM lookup specified ISDX to be used as G_IDX.	1	x	x
G_IDX	Generic index used to bind together entries across VCAP CLM lookups. G_IDX is calculated based on fields in VCAP CLM action from previous VCAP CLM lookup: NXT_IDX_CTRL. NXT_IDX. Default value is configurable in ANA_CL::CLM_MISC_CTRL.CLM_GIDX_DEF_SEL. It can be zero, logical port number, or masqueraded port number.	12	x	x
IGR_PORT_MASK_SEL	Mode selector for IGR_PORT_MASK. 0: Default setting. 1: Set for frames received from a loopback device, for example, LBK_DEV*. 2: Set for masqueraded frames if ANA_CL::CLM_MISC_CTRL.MASQ_IGR_MASK_ENA == 1. A masqueraded frame is identified by the following criteria: (ifh.fwd.dst_mode == inject && ifh.src_port != physical_src_port) || (misc.pipeline_act == inj_masq) 3: Set for the following frame types: 3.a: CPU injected frames and ANA_CL::CLM_MISC_CTRL.CPU_IGR_MASK_ENA == 1 3.b: Frames received from VD0 or VD1 and ANA_CL::CLM_MISC_CTRL.VD_IGR_MASK_ENA == 1 3.c: Frame received on a loopback device and ANA_CL::CLM_MISC_CTRL.LBK_IGR_MASK_SEL3_ENA == 1. If a frame fulfills multiple of above criteria, then higher value of IGR_PORT_MASK_SEL takes precedence.	2	x	

Table 49 • **VCAP CLM X16 Key Details *(continued)***

Field Name	Description	Size	NORMAL_7TUPLE	CUSTOM_1
IGR_PORT_MASK	Ingress port mask. IGR_PORT_MASK_SEL == 0: Each bit in the mask correspond to physical ingress port. IGR_PORT_MASK_SEL == 1: Each bit in the mask correspond to the physical port which the frame was looped on. IGR_PORT_MASK_SEL == 2: Each bit in the mask correspond to the masqueraded port. If IGR_PORT_MASK_SEL == 3: Bit 0: Physical src port == CPU0. Bit 1: Physical src port == CPU1. Bit 2: Physical src port == VD0. Bit 3: Physical src port == VD1. Bits 4:9: Src port (possibly masqueraded). Bits 44:48: ifh.misc.pipeline_pt. Bits 49:51: ifh.misc.pipeline_act. Bits 52: Reserved.	53	x	
L2_MC	Set if frame's destination MAC address is a multicast address (bit 40 = 1).	1	x	
L2_BC	Set if frame's destination MAC address is the broadcast address (FF-FF-FF-FF-FF-FF).	1	x	
TPID0	Tag protocol identifier of the frame's first tag (outer tag): 0: Untagged. 1: 0x8100. 4: 0x88A8. 5: Custom value 1. 6: Custom value 2. 7: Custom value 3.	3	x	
PCP0	By default PCP from frame but it is selectable per lookup in VCAP CLM whether to use the current classified PCP instead (ANA_CL::ADV_CL_CFG.USE_CL_TCI0_ENA).	3	x	
DEI0	By default DEI from frame but it is selectable per lookup in VCAP CLM whether to use the current classified DEI instead (ANA_CL::ADV_CL_CFG.USE_CL_TCI0_ENA).	1	x	
VID0	By default VID from frame but it is selectable per lookup in VCAP CLM whether to use the current classified VID instead (ANA_CL::ADV_CL_CFG.USE_CL_TCI0_ENA). For untagged frames, VID0 is set to 0.	12	x	

Table 49 • **VCAP CLM X16 Key Details *(continued)***

Field Name	Description	Size	NORMAL_7TUPLE CUSTOM_1
TPID1	Tag protocol identifier of the frame's second tag (tag after outer tag): 0: No second tag. 1: 0x8100. 4: 0x88A8. 5: Custom value 1. 6: Custom value 2. 7: Custom value 3.	3	x
PCP1	Frame's PCP from second tag.	3	x
DEI1	Frame's DEI from second tag.	1	x
VID1	Frame's VID from second tag.	12	x
TPID2	Tag protocol identifier of the frame's third tag: 0: No third tag. 1: 0x8100. 4: 0x88A8. 5: Custom value 1. 6: Custom value 2. 7: Custom value 3	3	x
PCP2	Frame's PCP from third tag.	3	x
DEI2	Frame's DEI from third tag.	1	x
VID2	Frame's VID from third tag.	12	x
L2_DMAC	Destination MAC address.	48	x
L2_SMAC	Source MAC address.	48	x
IP_MC	IPv4 frames: Set if frame's destination IP address is an IPv4 multicast address (0xE /4). IPv6 frames: Set if frame's destination IP address is an IPv6 multicast address (0xFF /8).	1	x
ETYPE_LEN	Frame type flag indicating that the frame is EtherType encoded. Set if frame has EtherType >= 0x600.	1	x

Table 49 • **VCAP CLM X16 Key Details *(continued)***

Field Name	Description	Size	NORMAL_7TUPLE CUSTOM_1
ETYPE	Overloaded field for different frame types: LLC frames (ETYPE_LEN = 0 and IP_SNAP = 0): ETYPE = [DSAP, SSAP] SNAP frames (ETYPE_LEN = 0 and IP_SNAP = 1): ETYPE = PID[4:3] from SNAP header. TCP/UDP IPv4 or IPv6 frames (ETYPE_LEN = 1, IP_SNAP = 1, and TCP_UDP = 1): ETYPE = TCP/UDP destination port. Non-TCP/UDP IPv4 or IPv6 frames (ETYPE_LEN = 1, IP_SNAP = 1, and TCP_UDP = 0): ETYPE = IP protocol. Other frames (ETYPE_LEN = 1 and IP_SNAP = 0): ETYPE = Frame's EtherType.	16	x
IP_SNAP	Frame type flag indicating that the frame is either an IP frame or a SNAP frame. IP frames (ETYPE_LEN = 1): Set if (EtherType= 0x0800 and IP version = 4) or (ETYPE = 0x86DD and IP version = 6). SNAP frames (ETYPE_LEN = 0): Set if LLC header contains AA-AA-03.	1	x
IP4	Frame type flag indicating the frame is an IPv4 frame. Set if frame is IPv4 frame (EtherType = 0x800 and IP version = 4).	1	x
L3_FRAGMENT	Set if IPv4 frame is fragmented (More Fragments flag = 1 or Fragments Offset > 0).	1	x
L3_FRAG_OFS_GT0	Set if IPv4 frame is fragmented but not the first fragment (Fragments Offset > 0)	1	x
L3_OPTIONS	Set if IPv4 frame contains options (IP len > 5). IP options are not parsed.	1	x
L3_DSCP	By default DSCP from frame. Per match, selectable to be current classified DSCP (ANA_CL::ADV_CL_CFG.USE_CL_DSCP_ENA).	6	x
L3_IP6_DIP	Destination IP address. IPv4 frames (IP4=1): Bits 31:0: Destination IPv4 address.	128	x

Table 49 • **VCAP CLM X16 Key Details *(continued)***

Field Name	Description	Size	NORMAL_7TUPLE	CUSTOM_1
L3_IP6_SIP	Overloaded field for different frame types: LLC frames (ETYPE_LEN=0 and IP_SNAP=0): L3_IP6_SIP = [CTRL, PAYLOAD[0:14]] SNAP frames (ETYPE_LEN=0 and IP_SNAP=1): L3_IP6_SIP = [PID[2:0], PAYLOAD[0:12]] IPv4 frames (ETYPE_LEN=1, IP_SNAP=1, and IP4=1): L3_IP6_SIP[31:0] = source IPv4 address IPv6 frames (ETYPE_LEN=1, IP_SNAP=1, IP4=0): L3_IP6_SIP = source IPv6 address Other frames (ETYPE_LEN=1, IP_SNAP=0): L3_IP6_SIP = PAYLOAD[0:15]	128	x	
TCP_UDP	Frame type flag indicating the frame is a TCP or UDP frame. Set if frame is IPv4/IPv6 TCP or UDP frame (IP protocol/next header equals 6 or 17). Overloading: Set to 1 for OAM Y.1731 frames.	1	x	
TCP	Frame type flag indicating the frame is a TCP frame. Set if frame is IPv4 TCP frame (IP protocol = 6) or IPv6 TCP frames (Next header = 6).	1	x	
L4_SPORT	TCP/UDP source port. Overloading: OAM Y.1731 frames: L4_SPORT = OAM MEL flags, see below. TCP_UDP is at the same time set to 1 to indicate the overloading. OAM MEL flags: Encoding of MD level/MEG level (MEL). One bit for each level where lowest level encoded as zero. The following keys can be generated: MEL=0: 0b0000000 MEL=1: 0b0000001 MEL=2: 0b0000011 MEL=3: 0b0000111 MEL=4: 0b0001111 MEL=5: 0b0011111 MEL=6: 0b0111111 MEL=7: 0b1111111 Together with the mask, the following kinds of rules may be created: Exact match. Fx. MEL=2: 0b0000011 Below. Fx. MEL<=4: 0b000XXXX Above. Fx. MEL>=5: 0bXX11111 Between. Fx. 3<= MEL<=5: 0b00XX111, where ‘X’ means don’t care.	16	x	

Table 49 • **VCAP CLM X16 Key Details *(continued)***

Field Name	Description	Size	NORMAL_7TUPLE	CUSTOM_1
L4_RNG	Range mask. Range types: SPORT, DPORT, SPORT or DPORT, VID, DSCP, custom. Input into range checkers is taken from frame, except DSCP value, which is the remapped DSCP value from basic classification.	8	x	
CUSTOM1	64 bytes payload starting from current frame pointer position, as controlled by VCAP CLM actions. Note that if frame_type==ETH, payload is retrieved from a position following the Ethernet layer. For example, after DMAC, SMAC, 0-3 VLAN tags, and EtherType.	512		x

3.13.7 VCAP CLM Actions

VCAP CLM supports four different actions, which can be used for different purposes. The actions have different sizes and must be paired with the keys as listed in the following table.

Table 50 • **VCAP CLM Action Selection**

Action Name	Size	Action Type
MLBS_REDUCED	1 word 69 bits	X1 type
CLASSIFICATION	2 words 124 bits	X2 type
MLBS	2 words 127 bits	X2 type
FULL	4 words 250 bits	X4 type

Any VCAP CLM action can be used with any of VCAP CLM keys. However, if the action's type is larger than the associated key's type (for instance FULL action of type X4 with TRI_VID key of type X8), then the entry row in the VCAP cannot be fully utilized.

The following table provides details for all VCAP CLM actions. When programming an action in VCAP CLM, the associated action fields listed must be programmed in the listed order with the first field in the table starting at bit 0 in the action.

Table 51 • **VCAP CLM Actions**

Field Name	Description and Encoding	Size	MLBS	MLBS_REDUCED	CLASSIFICATION	FULL
X2_TYPE	X2 type. 0: MLBS. 1: Classification.	1	x		x	

Table 51 • **VCAP CLM Actions *(continued)***

Field Name	Description and Encoding	Size	MLBS	MLBS_REDUCED	CLASSIFICATION	FULL
DSCP_ENA	If set, use DSCP_VAL as classified DSCP value.	1				x
DSCP_VAL	See DSCP_ENA.	6				x
COSID_ENA	If set, use COSID_VAL as classified COSID value.	1	x	x	x	x
COSID_VAL	See COSID_ENA.	3	x	x	x	x
QOS_ENA	If set, use QOS_VAL as classified QoS class.	1	x	x	x	x
QOS_VAL	See QOS_ENA.	3	x	x	x	x
DP_ENA	If set, use DP_VAL as classified drop precedence level.	1	x	x	x	x
DP_VAL	See DP_ENA.	2	x	x	x	x
DEI_ENA	If set, use DEI_VAL as classified DEI value.	1			x	x
DEI_VAL	See DEI_ENA.	1			x	x
PCP_ENA	If set, use PCP_VAL as classified PCP value.	1			x	x
PCP_VAL	See PCP_ENA.	3			x	x
MAP_LOOKUP_SEL	Selects which of the two QoS Mapping Table lookups that MAP_KEY and MAP_IDX are applied to. 0: No changes to the QoS Mapping Table lookup. That is, MAP_KEY and MAP_IDX are not used. 1: Update key type and index for QoS Mapping Table lookup #0. 2: Update key type and index for QoS Mapping Table lookup #1. 3: Reserved. MLBS_REDUCED: Use 8 bits from COSID_ENA (LSB), COSID_VAL, QOS_ENA, and QOS_VAL (MSB) as MAP_IDX. COSID and QOS class cannot be assigned at the same time. MAP_KEY is always 3 (TC).	2	x	x	x	x
MAP_KEY	Key type for QoS mapping table lookup. 0: DEI0, PCP0 (outer tag). 1: DEI1, PCP1 (middle tag). 2: DEI2, PCP2 (inner tag). 3: TC based on conf.mpls_sel_tc_only_ena it is either possible to always IFH.ENCAP.MPLS_TC or to only use TC it extracted by label stack. 4: PCP0 (outer tag). 5: Reserved. 6: DSCP if available, otherwise none (64 entries). 7: DSCP if available, otherwise DEI0, PCP0 (outer tag) if available using MAP_IDX+8, otherwise none (80 entries).	3	x		x	x
MAP_IDX	Index for QoS mapping table lookup. Index bits 10:3 into 2K mapping table. Bits 2:0 are always 0.	8	x		x	x

Table 51 • **VCAP CLM Actions *(continued)***

Field Name	Description and Encoding	Size	MLBS	MLBS_REDUCED	CLASSIFICATION	FULL
VID_ADD_REPLACE_SEL	Controls the classified VID. 0: New VID = old VID + VID_VAL. 1: New VID = VID_VAL. 2: New VID = VID from frame's second tag (middle tag) if available, otherwise VID_VAL. 3: New VID = VID from frame's third tag (inner tag) if available, otherwise VID_VAL.	2	x		x	x
VID_VAL	By default, add this value to the current classified VID. Note this is not the VID used as key into VCAP CLM. If the classified VID + VID_ADD_VAL exceeds the range of 12 bit then a wrap around is done.	12	x		x	x
VLAN_POP_CNT_ENA	If set, VLAN_POP_CNT is used.	1			x	x
VLAN_POP_CNT	VLAN pop count: 0, 1, 2, or 3 tags. Post-processing of VLAN_POP_CNT: If VLAN_POP_CNT exceeds the actual number of tags in the frame, it is reduced to match the number of VLAN tags in the frame.	2			x	x
VLAN_WAS_TAGGED	Controls the WAS_TAGGED setting forwarded to the rewriter. 0: No changes to WAS_TAGGED. 1: Set WAS_TAGGED to 0. 2: Set WAS_TAGGED to 1. 3: Reserved.	2			x	x
ISDX_ADD_REPLACE_SEL	Controls the classified ISDX. 0: New ISDX = old ISDX + ISDX_VAL. 1: New ISDX = ISDX_VAL.	1	x	x	x	x
ISDX_VAL	See ISDX_ADD_REPLACE_SEL.	9	x	x	x	x
MASK_MODE	Controls how PORT_MASK is applied. 0: OR_DSTMASK: Or PORT_MASK with destination mask. 1: AND_VLANMASK: And PORT_MASK with VLAN mask. The actual ANDing with the VLAN mask is performed after the ANA_L3 block has determined the VLAN mask. 2: REPLACE_PGID: Replace PGID port mask from MAC table lookup by PORT_MASK. 3: REPLACE_ALL: Use PORT_MASK as final destination set replacing all other port masks. 4: REDIR_PGID: Redirect using PORT_MASK[7:0] as PGID table index. See PORT_MASK for extra configuration options. 5: OR_PGID_MASK: Or PORT_MASK with PGID port mask from MAC table lookup. 6: Reserved. 7: Not applicable. Note that for REDIR_PGID, REPLACE_ALL, and VSTAX, the port mask becomes “sticky” and cannot be modified by subsequent processing steps. The CPU port is untouched by MASK_MODE.	3				x

Table 51 • **VCAP CLM Actions *(continued)***

Field Name	Description and Encoding	Size	MLBS	MLBS_REDUCED	CLASSIFICATION	FULL
PORT_MASK	Port mask. MASK_MODE=4 (REDIR_PGID): PORT_MASK[52]: SRC_PORT_MASK_ENA. If set, SRC_PORT_MASK is AND'ed with destination result. PORT_MASK[51]: AGGR_PORT_MASK_ENA. If set, AGGR_PORT_MASK is AND'ed with destination result. PORT_MASK[50]: VLAN_PORT_MASK_ENA. If set, VLAN_PORT_MASK is AND'ed with destination result. PORT_MASK[7:0]: PGID table index.	53				x
RT_SEL	Controls routing. 0: No change to routing. 1: Enable routing. 2: Disable routing.	2				x
FWD_DIS	If set, forwarding of the frame to front ports is disabled. CPU extraction is still possible.	1	x	x	x	
CPU_ENA	Copy frame to specified CPU extraction queue.	1	x	x	x	x
CPU_Q	CPU extraction queue when frame is forwarded to CPU.	3	x	x	x	x
MIP_SEL	Controls the MIP selection. 0: No change in MIP selection. 1: Enable MIP. 2: Disable MIP. If enabled, the MIP_IDX is given by the IPT table. OAM_Y1731_SEL determines the number of VLAN tags required for processing a frame in the MIP.	2			x	x
OAM_Y1731_SEL	Selects which frames are detected as OAM frames by MEP and MIP. 0: No change in detection. 1: Disable OAM. 2: Detect untagged OAM. 3: Detect single tagged OAM. 4: Detect double tagged OAM. 5: Detect triple tagged OAM. 6: Enable Y1731 OAM unconditionally. 7: Enable any tags and EtherType as OAM unconditionally.	3	x		x	x
RSVD_LBL_VAL	Use when MPLS_OAM_TYPE = 2, 4, or 5. Configures the reserved label used as OAM label for this MEP or MIP. Only the lower 4 bits of the label value is configured. Normally this field should be set to 13 (GAL label value). If MPLS_OAM_TYPE is set to 2, 4, or 5 and action record does not include RSVD_LBL_VAL, then a value of 13 (GAL label value) is used.	4				x
TC_ENA	Use the label's TC as classified TC.	1		x		
TTL_ENA	Use the label's TTL as classified TTL	1		x		

Table 51 • **VCAP CLM Actions *(continued)***

Field Name	Description and Encoding	Size	MLBS	MLBS_REDUCED CLASSIFICATION	FULL
TC_LABEL	Selects which label provides the classified TC. 0: Use TC from LSE #0 (if available). 1: Use TC from LSE #1 (if available). 2: Use TC from LSE #2 (if available). 3: Do not update TC. LSE #0 is top LSE, followed by LSE #1, and so on.	2	x		x
TTL_LABEL	Selects which label provides the classified TTL. 0: Use TTL from LSE #0 (if available). 1: Use TTL from LSE #1 (if available). 2: Use TTL from LSE #2 (if available). 3: Do not update TTL. LSE #0 is top LSE, followed by LSE #1, and so on.	2	x		x
NUM_VLD_LABELS	Number of valid labels, without reserved labels. Position of "deepest" label to be processed. Numbering starts with 0 at top LSE. If reserved labels are "skipped", then these are not counted in NUM_VLD_LABELS. Valid range: 0-2. Use of NUM_VLD_LABELS is dependent on FWD_TYPE as follows: FWD_TYPE=0: NUM_VLD_LABELS is not used. FWD_TYPE=1: Position of PW label. FWD_TYPE=2: Position of label to be swapped. LSEs above the swap label are popped. FWD_TYPE=3: Position of deepest label, which is to be popped.	2	x		x
FWD_TYPE	Forwarding type. 0: NONE 1: TERMINATE_PW - Terminate pseudo wire. 2: LBL_SWAP - Swap label 3: LBL_POP - Pop Labels 3: CTRL_PDU	4	x	x	x
	FWD_TYPE: NONE No MPLS termination, MPLS SWAP, or MPLS POP				
	FWD_TYPE: TERMINATE_PW Terminate pseudo wire. NXT_NORM_W16_OFFSET or NXT_NORM_W32_OFFSET must be set to move frame pointer to CW/ACH if present, and otherwise to the PDU that follows the MPLS label stack. NUM_VLD_LABELS must be set to the position of the PW label. NXT_TYPE_AFTER_OFFSET must be set to CW if PW uses CW, otherwise ETH. NXT_NORMALIZE must be set to 1 to have MPLS Link Layer, LSEs and CW stripped. MPLS_OAM_TYPE can be set to values 0-4 to disable/enable OAM PDU detection.				

Table 51 • **VCAP CLM Actions *(continued)***

Field Name	Description and Encoding	Size	MLBS	MLBS_REDUCED	CLASSIFICATION	FULL
	FWD_TYPE: LBL_SWAP Swap label at position NUM_VLD_LABELS. Any label above the swap label are popped. NXT_NORM_W16_OFFSET or NXT_NORM_W32_OFFSET must be set to move frame pointer to swap label. NUM_VLD_LABELS must be set to the position of the swap label. NXT_NORMALIZE must be set to 1 to have MPLS Link Layer and, possibly, labels stripped. NXT_TYPE_AFTER_OFFSET must be set to MPLS. OAM PDUs are not detected, though frames with reserved label values as well as frames with TTL expiry can be redirected to CPU.					
	FWD_TYPE: LBL_POP Pop labels. NXT_NORM_W16_OFFSET or NXT_NORM_W32_OFFSET must be set to move frame pointer past popped labels. NUM_VLD_LABELS must be set to the position of the deepest label to be popped. NXT_NORMALIZE must be set to 1. NXT_TYPE_AFTER_OFFSET should normally be set to MPLS. NXT_TYPE_AFTER_OFFSET may be set to CW when terminating LSP with IP frames for CPU processing. MPLS_OAM_TYPE can be set to 0, 5, or 6 to disable/enable OAM PDU detection.					
	FWD_TYPE: CTRL_PDU MPLS_OAM_TYPE controls PDU type (as encoded in IFH.ENCAP.PDU_TYPE): 0: NONE. 1: OAM_Y1731. 2: OAM_MPLS_TP. 3: PTP. 4: IP4_UDP_PTP. 5: IP6_UDP_PTP. 6: Reserved. 7: SAM_SEQ. OAM_Y1731_SEL controls PDU_OFFSET: 0: No change to PDU offset. 1: ETH_PAYLOAD. 2: IP_PAYLOAD. 3: Non-normalized offset (PDU_OFFSET = difference between normalized and non-normalized payload).					

Table 51 • **VCAP CLM Actions *(continued)***

Field Name	Description and Encoding	Size	MLBS	MLBS_REDUCED CLASSIFICATION	FULL
MPLS_OAM_TYPE	When either MIP or MEP OAM detection is enabled (MPLS_MIP_ENA or MPLS_MEP_ENA), MPLS_OAM_TYPE configures in which control channel to look for OAM. For FWD_TYPE = 1 (TERMINATE_PW), MPLS_OAM_TYPE configures which Control Channel Type is used for forwarding PW OAM: 1: Detect Vccv1 OAM 2: Detect Vccv2 OAM 3: Detect Vccv3 OAM 4: Detect Vccv4 OAM For FWD_TYPE = 3 (LSP_POP), MPLS_OAM_TYPE configures which Control Channel Type is used for LSP OAM: 5: LSP OAM under GAL found at pos NUM_VLD_LABELS + 1 6: LSP OAM under GAL found at pos NUM_VLD_LABELS For FWD_TYPE = 4 (CTRL_PDU), MPLS_OAM_TYPE configures directly the control type and PDU position (with OAM_Y1731_SEL and NXT_NORM_W16_OFFSET/ NXT_NORM_W32_OFFSET) and bypasses post-processing: 1: OAM_Y1731 2: OAM_MPLS_TP 3: PTP 4: IP4_UDP_PTP 5: IP6_UDP_PTP 6: Reserved 7: SAM_SEQ	3	x	x	x
MPLS_MEP_ENA	Enables detection of MPLS MEP. If enabled and if frame is not selected for MIP extraction, it is determined whether it is a valid frame OAM PDU according to the MPLS_OAM_TYPE.	1	x	x	x
MPLS_MIP_ENA	Enables detection of MPLS MIP. If enabled, the iTTL will be checked for expiry. In case of iTTL expiry, the frame is checked if it is a valid OAM PDU according to the MPLS_OAM_TYPE.	1	x	x	x
MPLS_OAM_FLAVOR	If OAM detection is enabled, this setting determines if the G-ACH/ACH selected by MPLS_OAM_TYPE is checked for Channel Type = 0x8902. This configuration applies only to MEP OAM detection. OAM MIP detection is extracted to SW based on from the configured Control Channel Type, regardless of the channel type, because all OAM MIP processing is done in software. 0: G8113_2 (BFD, and so on). 1: G8113_1 (BHH).	1	x	x	x
MPLS_IP_CTRL_ENA	Enables IP directly under MPLS based on IP profiles.	1	x	x	x

Table 51 • **VCAP CLM Actions *(continued)***

Field Name	Description and Encoding	Size	MLBS	MLBS_REDUCED	CLASSIFICATION	FULL
CUSTOM_ACE_ENA	Controls S2 custom rule selection: Bit 0: Selects custom key to use (0: CUSTOM_1, 1: CUSTOM_2). Bit 1: Enables custom key for first lookup. Bit 2: Enables custom key for second lookup.	3				x
CUSTOM_ACE_OFFSET	Selects custom data extraction point: 0: Current position (after IFH.WORD16_POP_CNT). Only applicable to frame type ETH. 1: Link layer payload (after VLAN tags and EtherType for ETH, after control word for CW). 2: Link layer payload + 20 bytes. 3: Link layer payload + 40 bytes.	2				x
PAG_OVERRIDE_MASK	Bits set in this mask override the previous PAG result with the PAG_VAL value from this action.	8	x		x	x
PAG_VAL	PAG is updated using the following bit mask operation: PAG = (PAG and ~PAG_OVERRIDE_MASK) | (PAG_VAL and PAG_OVERRIDE_MASK). PAG can be used to tie VCAP CLM lookups with VCAP IS2 lookups. The PAG default value is configurable per port in ANA_CL:PORT:PORT_ID_CFG.PAG_VAL.	8	x		x	x
RESERVED	Must be set to 0.	7				x
LPORT_NUM	Use this value for LPORT and IFH.SRC_PORT when LPORT_ENA is set.	6				x
MATCH_ID	Logical ID for the entry. If the frame is forwarded to the CPU, the MATCH_ID is extracted together with the frame. MATCH_ID is used together with MATCH_ID_MASK as follows: MATCH_ID = (MATCH_ID and ~MATCH_ID_MASK) | (MATCH_ID and MATCH_ID_MASK). The result is placed in IFH.CL_RSLT.	12				x
MATCH_ID_MASK	Bits set in this mask overrides the previous MATCH_ID with the MATCH_ID value from this action.	12				x
PIPELINE_FORCE_ENA	If set, use PIPELINE_PT unconditionally. Overrule previous settings of pipeline point. Use the following pipeline actions: 0: No change to PIPELINE_PT or PIPELINE_ACT. 1: Change PIPELINE_PT and set PIPELINE_ACT=XTR 2: Change PIPELINE_PT and set PIPELINE_ACT=INJ (PIPELINE_ACT_SEL=0) or INJ_MASQ (PIPELINE_ACT_SEL=1) 3: Change PIPELINE_PT and set PIPELINE_ACT=LBK_QS (PIPELINE_ACT_SEL=0) or LBK_ASM (PIPELINE_ACT_SEL=1)	2	x	x	x	x

Table 51 • **VCAP CLM Actions *(continued)***

Field Name	Description and Encoding	Size	MLBS	MLBS_REDUCED	CLASSIFICATION	FULL
PIPELINE_PT_REDUCED	Pipeline point used if PIPELINE_FORCE_ENA is set: 0: ANA_CLM 1: ANA_OU_MIP 2: ANA_IN_MIP 3: ANA_PORT_VOE 4: ANA_OU_VOE 5: ANA_IN_VOE 6: REW_IN_VOE 7: REW_OU_VOE	3		x		
PIPELINE_PT	Pipeline point used if PIPELINE_FORCE_ENA is set: 0: NONE 1: ANA_VRAP 2: ANA_PORT_VOE 3: ANA_CL 4: ANA_CLM 5: ANA_IPT_PROT 6: ANA_OU_MIP 7: ANA_OU_SW 8: ANA_OU_PROT 9: ANA_OU_VOE 10: ANA_MID_PROT 11: ANA_IN_VOE 12: ANA_IN_PROT 13: ANA_IN_SW 14: ANA_IN_MIP 15: ANA_VLAN 16: ANA_DONE	5	x		x	x

Table 51 • **VCAP CLM Actions *(continued)***

Field Name	Description and Encoding	Size	MLBS	MLBS_REDUCED	CLASSIFICATION	FULL
NXT_KEY_TYPE	Enforces specific key type for next VCAP CLM lookup. 0: No overruling of default key type selection. 1: MLL 2: SGL_MLBS 3: DBL_MLBS 4: TRI_MLBS 5: TRI_VID or TRI_VID_IDX depending on configuration per VCAP CLM lookup in ANA_CL::CLM_KEY_CFG.CLM_TRI_VID_SEL 6: LL_FULL 7: NORMAL (with normal SRC information) 8: NORMAL with DST information This is a modified version of NORMAL key type, where - the frame's DMAC is used in the L2_SMAC key field. - the frame's IPv4 DIP is used in the L3_IP4_SIP key field. 9: NORMAL_7TUPLE 10: NORMAL_5TUPLE_IP4 11: PURE_5TUPLE_IP4 12: CUSTOM1 13: CUSTOM2 14: CUSTOM4 15: CLMS_DONE Complete - disable all remaining VCAP CLM lookups.	4	x	x	x	x
NXT_NORM_W32_OFFSET	4-byte value to be added to frame pointer. If any reserved MPLS labels were "skipped" during key generation before VCAP CLM lookup, then these are not counted in NXT_NORM_W32_OFFSET, because frame pointer is automatically moved past "skipped" labels. If both NXT_NORM_W32_OFFSET and NXT_OFFSET_FROM_TYPE are specified, then frame pointer is first modified based on NXT_OFFSET_FROM_TYPE and then moved further based on NXT_NORM_W32_OFFSET.	2		x		
NXT_NORM_W16_OFFSET	2-byte value to be added to frame pointer. If any reserved MPLS labels were skipped during key generation before VCAP CLM lookup, then these are not be counted in NXT_NORM_W16_OFFSET, because frame pointer is automatically moved past "skipped" labels. If both NXT_NORM_W16_OFFSET and NXT_OFFSET_FROM_TYPE are specified, then frame pointer is first modified based on NXT_OFFSET_FROM_TYPE and then moved further based on NXT_NORM_W16_OFFSET.	5	x		x	x

Table 51 • **VCAP CLM Actions *(continued)***

Field Name	Description and Encoding	Size	MLBS	MLBS_REDUCED	CLASSIFICATION	FULL
NXT_OFFSET_FROM_TYPE	Move frame pointer depending on type of current protocol layer. 0: NONE. No frame pointer movement. 1: SKIP_ETH. Skip Ethernet layer. If frame type is ETH, then move frame pointer past DMAC, SMAC and 0-3 VLAN tags. 2: SKIP_IP_ETH - Skip IP layer and preceding Ethernet layer (if present). If frame type is ETH: Move frame pointer past DMAC, SMAC, 0-3 VLAN tags and IPv4/IPv6 header. If frame type is CW and IP version is 4 or 6: Move frame pointer past IPv4/IPv6 header. For IPv4 header frame pointer is also moved past any IPv4 header options. 3: RSV. Reserved.	2	x		x	x
NXT_TYPE_AFTER_OFFSET	Protocol layer (frame_type) at frame pointer position after update based on NXT_OFFSET_FROM_TYPE and NXT_NORM_W16_OFFSET or NXT_NORM_W32_OFFSET. Frame_type is only changed if one of the following conditions is true: a) Frame pointer is moved. b) Frame_type going into VCAP CLM was DATA and NXT_KEY_TYPE != 0. 0: ETH. Frame pointer points to start of DMAC. 1: CW (IP / MPLS PW CW / MPLS ACH). Frame pointer points to MPLS CW/ACH or IP version. 2: MPLS. Frame pointer points to MPLS label. 3: DATA. “Raw” data, such as unknown protocol type.	2	x	x	x	x
NXT_NORMALIZE	Instruct rewriter to strip all frame data up to current position of frame pointer. The stripped data is not used for forwarding. When a VCAP CLM action is processed, frame pointer is updated before NXT_NORMALIZE is applied. Note The rewriter can strip a maximum of 42 bytes.	1	x	x	x	x

Table 51 • **VCAP CLM Actions *(continued)***

Field Name	Description and Encoding	Size	MLBS	MLBS_REDUCED	CLASSIFICATION	FULL
NXT_IDX_CTRL	Controls the generation of the G_IDX used in the VCAP CLM next lookup: 0: TRANSPARENT. G_IDX of previous parser step is carried forward to next parser step. 1: REPLACE. Replace previous G_IDX value with NXT_IDX of this result. 2: ADD. Add NXT_IDX of this result to G_IDX of previous parser step. 3: ISDX.Use ISDX as G_IDX. 4: RPL_COND_ISDX. Replace previous G_IDX value with NXT_IDX of this result. Assign ISDX to ISDX_VAL for terminated data. 5: Use ISDX with NXT_IDX for non terminated data (not CPU redir and MEP traffic). 6: Replace ISDX with NXT_IDX. Use ISDX as G_IDX in next key for non terminated data. 7: Reserved.	3	x	x	x	x
NXT_IDX	Index used as part of key (field G_IDX) in the next lookup.	12	x	x	x	x
RESERVED	Must be set to 0.	1	x		x	x
PIPELINE_ACT_SEL	Used by PIPELINE_FORCE_ENA to control PIPELINE_ACT, see PIPELINE_FORCE_ENA.	1	x		x	x
LPORT_ENA	Controls if LPORT_NUM updates LPORT and IFH.SRC_PORT.	1				x

3.14 Analyzer Classifier

The analyzer classifier (ANA_CL) handles frame classification. This section provides information about the following tasks performed by ANA_CL.

- Basic classification. Initial classification including frame filtering, VLAN and QoS classification.
- VCAP CLM processing. Advanced classification including MPLS classification, service classification, and OAM detection.
- QoS mappings. Mapping and translations of incoming QoS parameters.
- Ingress protection. Mapping to a group protection for fast failover.
- Layer 2 control protocol processing. Tunneling, peer, discard of L2CP frames per EVC.
- Ethernet Y.1731 MIP processing. Down-MIP half function with CPU extraction of CCM, LTM, and LBM frames.

3.14.1 Basic Classifier

The analyzer classifier includes a common basic classifier that determines the following basic properties that affect the forwarding of each frame through the switch.

- VLAN tag extraction. Parses the frame's VLAN tags in accordance with known tag protocol identifier values.
- Pipeline point evaluation. For injected and looped frames, ensures ANA_CL is part of the frame's processing flow.
- Routing control. Evaluates if a frame can be routed in terms of VLAN tags.
- Frame acceptance filtering. Drops illegal frame types.
- QoS classification. Assigns one of eight QoS classes to the frame.

- Drop precedence (DP) classification. Assigns one of four drop precedence levels to the frame.
- DSCP classification. Assigns one of 64 DSCP values to the frame.
- VLAN classification. Extracts tag information from the frame or use the port VLAN.
- Link aggregation code generation. Generates the link aggregation code.
- Layer 2 and Layer 3 control protocol handling. Determines CPU forwarding, CPU extraction queue number, and dedicated protocol QoS handling.
- Versatile Register Access Protocol (VRAP) handling. Extracts VRAP frames to VRAP engine.
- Logical port mapping. Maps physical ingress port to logical port used by analyzer.
- Port VOE VLAN awareness. Extracts PCP and DEI information from incoming frame used by port VOE in loss measurement counters.

The outcome of the classifier is the basic classification result, which can be overruled by more intelligent frame processing with VCAP classification matching (VCAP CLM).

3.14.1.1 VLAN Tag Extraction

The following table lists the register associated with VLAN tag extraction.

Table 52 • **VLAN Tag Extraction Register**

Register	Description	Replication
ANA_CL::VLAN_STAG_CFG	Ethertype for S-tags in addition to default value 0x88A8.	3

Up to five different VLAN tags are recognized by the device.

- Customer tags (C-tags), which use TPID 0x8100.
- Service tags (S-tags), which use TPID 0x88A8 (IEEE 802.1ad).
- Service tags (S-tags), which use a custom TPID programmed in ANA_CL::VLAN_STAG_CFG. Three different custom TPIDs are supported.

The device can parse and use information from up to three VLAN tags of any of the types previously described.

Various blocks in the device use Layer 3 and Layer 4 information for classification and forwarding. Layer 3 and Layer 4 information can be extracted from a frame with up to three VLAN tags. Frames with more than three VLAN tags are always considered non-IP frames.

3.14.1.2 Pipeline Point Evaluation

The basic classifier is active for all frames with no active pipeline information for frames injected before or at pipeline point ANA_CL and for frames extracted at or after pipeline point ANA_CL.

Frames looped by the rewriter (PIPELINE_ACT = LBK_ASM) have their REW pipeline point changed to an ANA pipeline point, which indicates the point in the analyzer flow in ANA where they appear again after being looped. The following pipeline point translation are handled.

- PIPELINE_PT = REW_PORT_VOE is changed to ANA_PORT_VOE
- PIPELINE_PT = REW_OU_VOE is changed to ANA_OU_VOE
- PIPELINE_PT = REW_IN_VOE is changed to ANA_IN_VOE
- PIPELINE_PT = REW_SAT is changed to ANA_CLM
- PIPELINE_PT = REW_OU_VOE is changed to ANA_OU_SW
- PIPELINE_PT = REW_IN_SW is changed to ANA_IN_SW

3.14.1.3 Routing Tag Control

The basic classifier determines which IP frames can be routed in terms of the VLAN tag types. If a frame can be routed, it is signalled to ANA_L3, where the routing decision is made. The following table lists the register associated with routing control.

Table 53 • **Routing Tag Control Register**

Register	Description	Replication
ANA_CL:PORT:VLAN_TPID_CTRL. BASIC_TPID_AWARE_DIS	Configures valid TPIDs for routing. This configuration is shared with VLAN classification.	Per port
ANA_CL:PORT:VLAN_TPID_CTRL. RT_TAG_CTRL	Configures number of valid VLAN tags in routable frames.	Per port

In order for a frame to be routed, the frame's VLAN tags must all be valid, and routing must be enabled for the number of VLAN tags in the frame.

The settings in BASIC_TPID_AWARE_DIS define each of three processed VLAN tags of which TPID values are valid for routing. The settings in RT_TAG_CTRL define the number of valid VLAN tags that must be present in routable frames.

Example: If routing is enabled for single-tagged frames with TPID=0x8100, configure the following:

- BASIC_TPID_AWARE_DIS = 0x7FFE. This invalidates all TPIDs except TPID = 0x8100 for outer most VLAN tag.
- RT_TAG_CTRL = 0x2. This enables routing of frames with one accepted tag only.

In this example, if an untagged frame is received, it is not routable because routing is not enabled for untagged frames. If a single-tagged frame is received with for instance TPID=0x88A8 or a double-tagged frame is received with for instance TPID=0x8100 for outer tag and TPID=0x8100 for inner tag, they are not routable because not all their VLAN tags are valid.

Example: To configure routing on a VLAN unaware port where routing is enabled for untagged frames only, set RT_TAG_CTRL = 0x1 to enable routing of untagged frames only.

3.14.1.4 Frame Acceptance Filtering

The following table lists the registers associated with frame acceptance filtering.

Table 54 • **Frame Acceptance Filtering Registers**

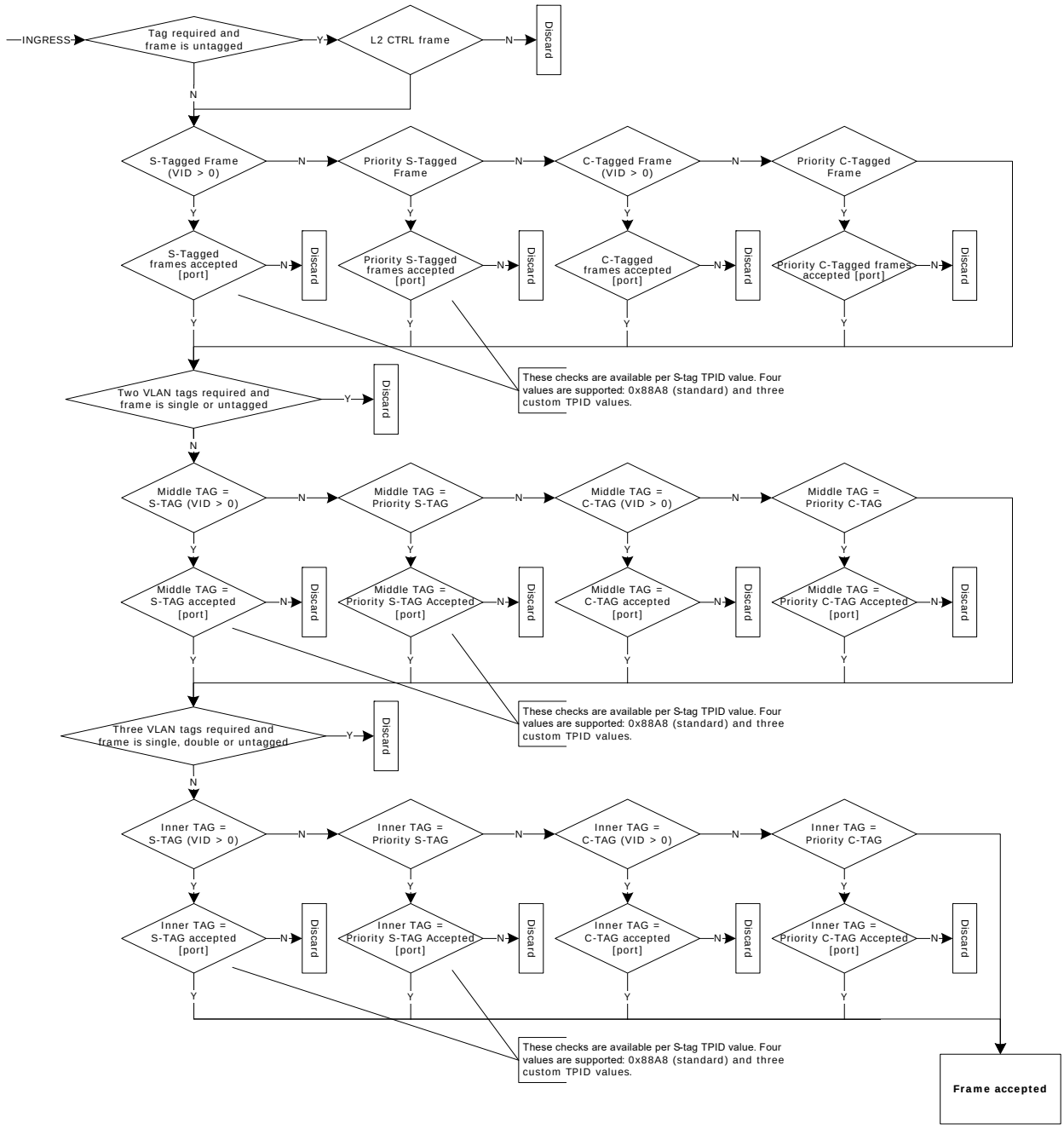
Register	Description	Replication
ANA_CL::FILTER_CTRL	Configures filtering of special frames	Per port
ANA_CL::VLAN_FILTER_CTRL	Configures VLAN acceptance filter	Three per port

Based on the configurations in the FILTER_CTRL register, the classifier can discard certain frame types that are normally not allowed to enter the network, such as the following.

- Frames with a multicast source MAC address (bit 40 in address is set)
- Frames with a null source or null destination MAC address (address = 0x000000000000)

The VLAN acceptance filter decides whether a frame's VLAN tagging is allowed on the port. It has three stages where each stage checks one of the up to three processed VLAN tags. Each stage is programmable independently of the other stages in ANA_CL:PORT:VLAN_FILTER_CTRL. The following illustration shows the flowchart for the VLAN acceptance filter.

Figure 30 • **VLAN Acceptance Filter**



The VLAN acceptance filter does not apply to untagged Layer 2 control protocol frames. They are always accepted even when VLAN tags are required for other frames. The following frames are recognized as Layer 2 control protocol frames:

- Frames with DMAC = 0x0180C2000000 through 0x0180C200000F
- Frames with DMAC = 0x0180C2000020 through 0x0180C200002F
- Frames with DMAC = 0x0180C2000030 through 0x0180C200003F

Tagged Layer 2 control protocol frames are handled by the VLAN acceptance filter like normal tagged frames and they must comply with the filter settings to be accepted.

Frames discarded by the frame acceptance filters are assigned PIPELINE_PT = ANA_CL and learning of the frames' source MAC addresses are at the same time disabled.

3.14.1.5 QoS, DP, and DSCP Classification

This section provides information about the functions in the QoS, DP, and DSCP classification. The three tasks are described as one, because the tasks have functionality in common.

The following table lists the registers associated with QoS, DP, and DSCP classification.

Table 55 • **QoS, DP, and DSCP Classification Registers**

Register	Description	Replication
ANA_CL:PORT:QOS_CFG	Configuration of the overall classification flow for QoS, DP, and DSCP.	Per port
ANA_CL:PORT:PCP_DEI_MAP_CFG	Port-based mapping of (DEI, PCP) to (DP, QoS).	Per port per DEI per PCP (16 per port)
ANA_CL::DSCP_CFG	DSCP configuration per DSCP value.	Per DSCP (64)
ANA_CL::QOS_MAP_CFG.DSCP_REWR_VAL	DSCP rewrite values per QoS class.	Per QoS class (8)
ANA_CL::CPU_8021_QOS_CFG	Configuration of QoS class for Layer 2 control protocol frames redirected to the CPU.	Per Layer 2 protocol address (32)

The basic classification provides the user with control of the QoS, DP, and DSCP classification algorithms. The result of the basic classification are the following frame properties:

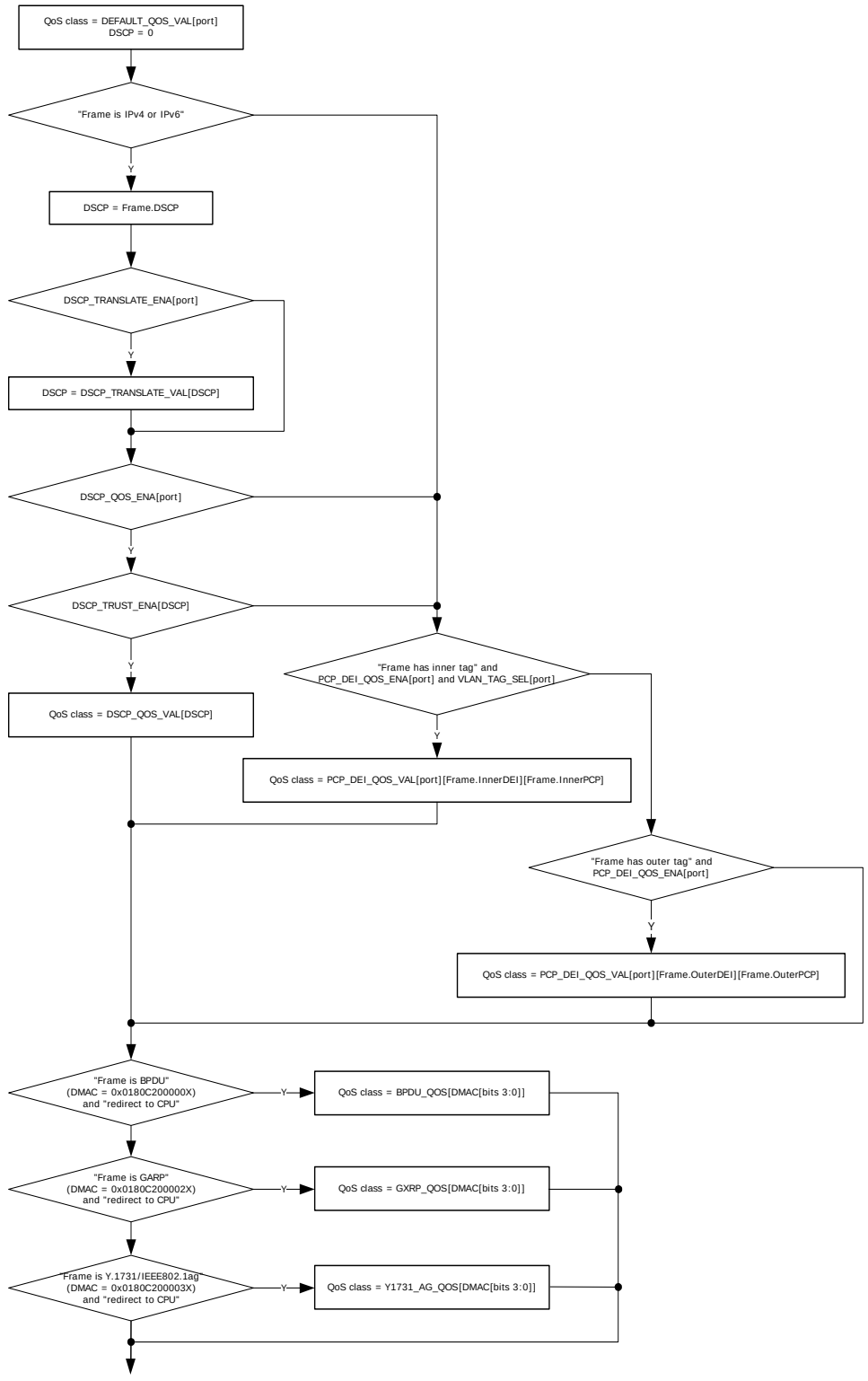
- The frame's QoS class. This class is encoded in a 3-bit field, where 7 is the highest priority QoS class and 0 is the lowest priority QoS class. The QoS class is used by the queue system when enqueuing frames and when evaluating resource consumptions, for policing, statistics, and rewriter actions. The QoS class is carried internally in the IFH field IFH.VSTAX.QOS.CL_QOS.
- The frame's DP level. This level is encoded in a 2-bit field, where frames with DP = 3 have the highest probability of being dropped and frames with DP = 0 have the lowest probability. The DP level is used by the MEF compliant policers for measuring committed and peak information rates, for restricting memory consumptions in the queue system, for collecting statistics, and for rewriting priority information in the rewriter. The DP level is incremented by the policers if a frame is exceeding a programmed committed information rate. The DP level is carried internally in the IFH field IFH.VSTAX.QOS.CL_DP.
- The frame's DSCP. This value is encoded in a 6-bit fields. The DSCP value is forwarded with the frame to the rewriter where it is translated and rewritten into the frame. The DSCP value is only applicable to IPv4 and IPv6 frames. The DSCP is carried internally in the IFH field IFH.QOS.DSCP.

The classifier looks for the following fields in the incoming frame to determine the QoS, DP, and DSCP classification:

- Priority Code Point (PCP) when the frame is VLAN tagged or priority tagged. There is an option to use the inner tag for double tagged frames (VLAN_CTRL.VLAN_TAG_SEL).
- Drop Eligible Indicator (DEI) when the frame is VLAN tagged or priority tagged. There is an option to use the inner tag for double tagged frames (VLAN_CTRL.VLAN_TAG_SEL).
- DSCP (all 6 bits, both for IPv4 and IPv6 packets). The classifier can look for the DSCP value behind up to three VLAN tags. For non-IP frames, the DSCP is 0 and it is not used elsewhere in the switch.
- Destination MAC address (DMAC) for L2CP frames. Layer 2 control protocol frames that are redirected to the CPU are given a programmable QoS class independently of other frame properties.

The following illustration shows the flow chart of basic QoS classification.

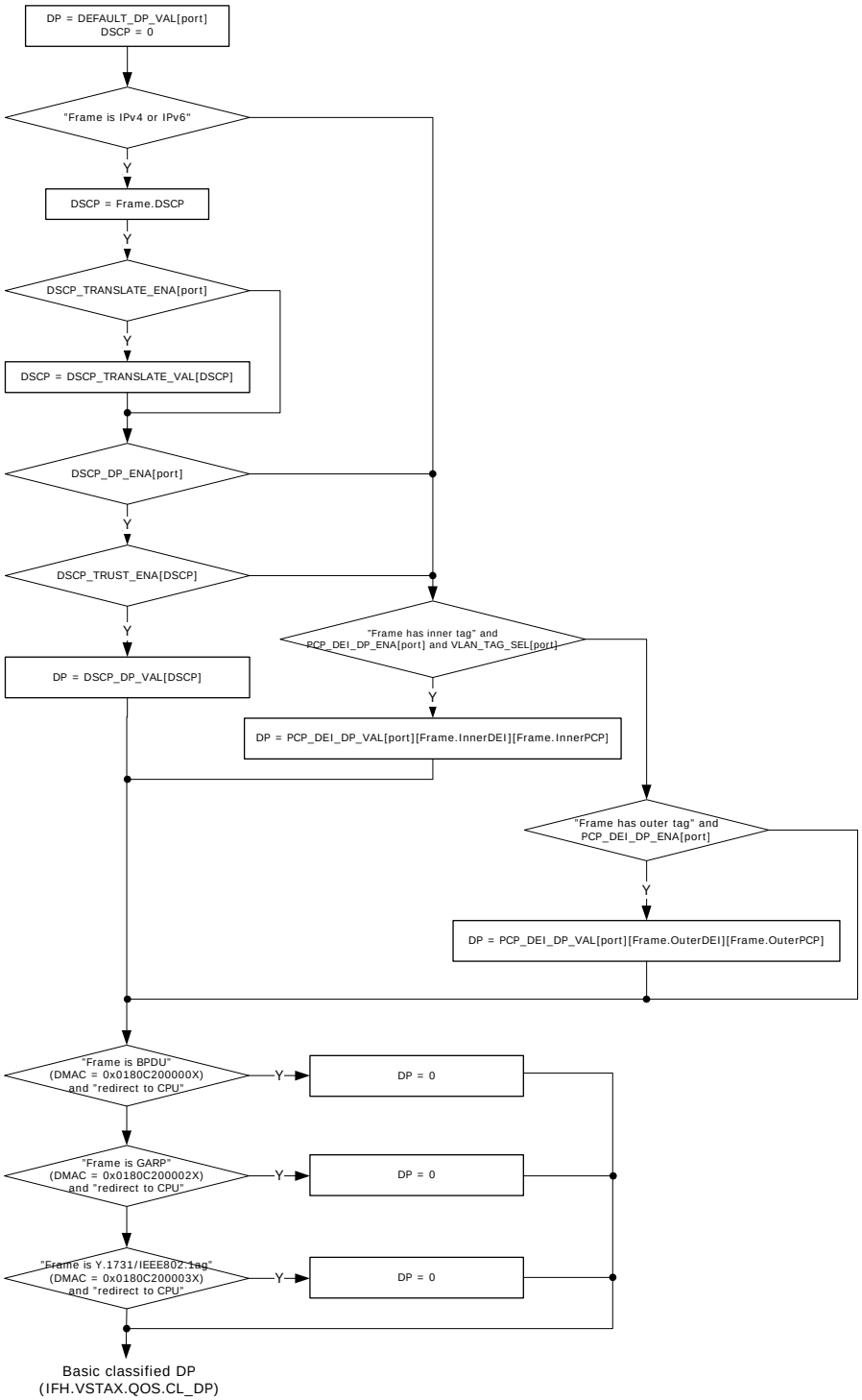
Figure 31 • **Basic QoS Classification Flow Chart**



Basic classified QoS class (IFH.VSTAX.QOS.CL_QOS)

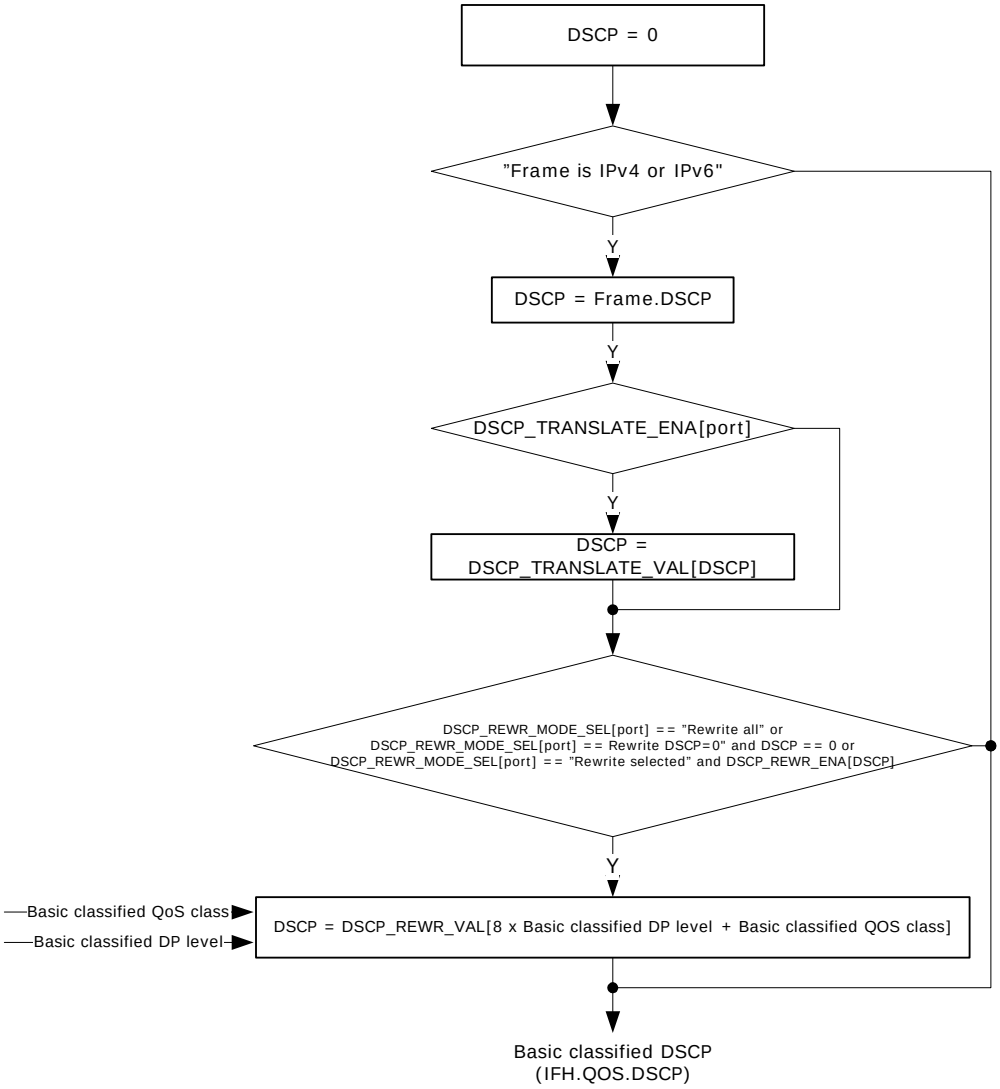
The following illustration shows the flow chart for basic DP classification.

Figure 32 • **Basic DP Classification Flow Chart**



The following illustration shows the flow chart for basic DSCP classification.

Figure 33 • **Basic DSCP Classification Flow Chart**



Note that the DSCP translation part is common for both QoS, DP, and DSCP classification, and the DSCP trust part is common for both QoS and DP classification.

The basic classifier has the option to overrule rewriter configuration (REW:PORT:DSCP_MAP) that remaps the DSCP value at egress. When ANA_CL:PORT:QOS_CFG.DSCP_KEEP_ENA is set, the rewriter is instructed not to modify the frame's DSCP value (IFH.QOS.TRANSP_DSCP is set to 1).

The basic classified QoS, DP, and DSCP can be overwritten by more intelligent decisions made in the VCAP CLM.

3.14.1.6 VLAN Classification

The following table lists the registers associated with VLAN classification.

Table 56 • **VLAN Configuration Registers**

Register	Description	Replication
ANA_CL:PORT:VLAN_CTRL	Configures the port's processing of VLAN information in VLAN-tagged and priority-tagged frames. Configures the port-based VLAN.	Per port

Table 56 • **VLAN Configuration Registers *(continued)***

ANA_CL::VLAN_STAG_CFG	Configures custom S-tag TPID value.	3
ANA_CL:PORT:PCP_DEI_TRANS_CFG	Port-based translation of (DEI, PCP) to basic classified (DEI, PCP).	Per port per DEI per PCP (16 per port)
ANA_CL:PORT:VLAN_TPID_CTRL.BASIC_TPID_AWARE_DIS	Configures valid TPIDs for VLAN classification. This configuration is shared with routing tag control.	Per port

The VLAN classification determines the following set of VLAN parameters for all frames:

- Priority Code Point (PCP). The PCP is carried internally in the IFH field IFH.VSTAX.TAG.CL_PCP.
- Drop Eligible Indicator (DEI). The DEI is carried internally in the IFH field IFH.VSTAX.TAG.CL_DEI.
- VLAN Identifier (VID). The VID is carried internally in the IFH field IFH.VSTAX.TAG.CL_VID.
- Tag Protocol Identifier (TPID) type, which holds information about the TPID of the tag used for classification. The TPID type is carried internally in IFH field IFH.VSTAX.TAG.TAG_TYPE. Extended information about the tag type is carried in IFH.DST.TAG_TPID.

The device recognizes five types of tags based on the TPID, which is the EtherType in front of the tag:

- Customer tags (C-tags), which use TPID 0x8100.
- Service tags (S-tags), which use TPID 0x88A8 (IEEE 802.1ad).
- Custom service tags (S-tags), which use one of three custom TPIDs programmed in ANA_CL::VLAN_STAG_CFG. Three different values are supported.

For customer tags and service tags, both VLAN tags (tags with nonzero VID) and priority tags (tags with VID = 0) are processed.

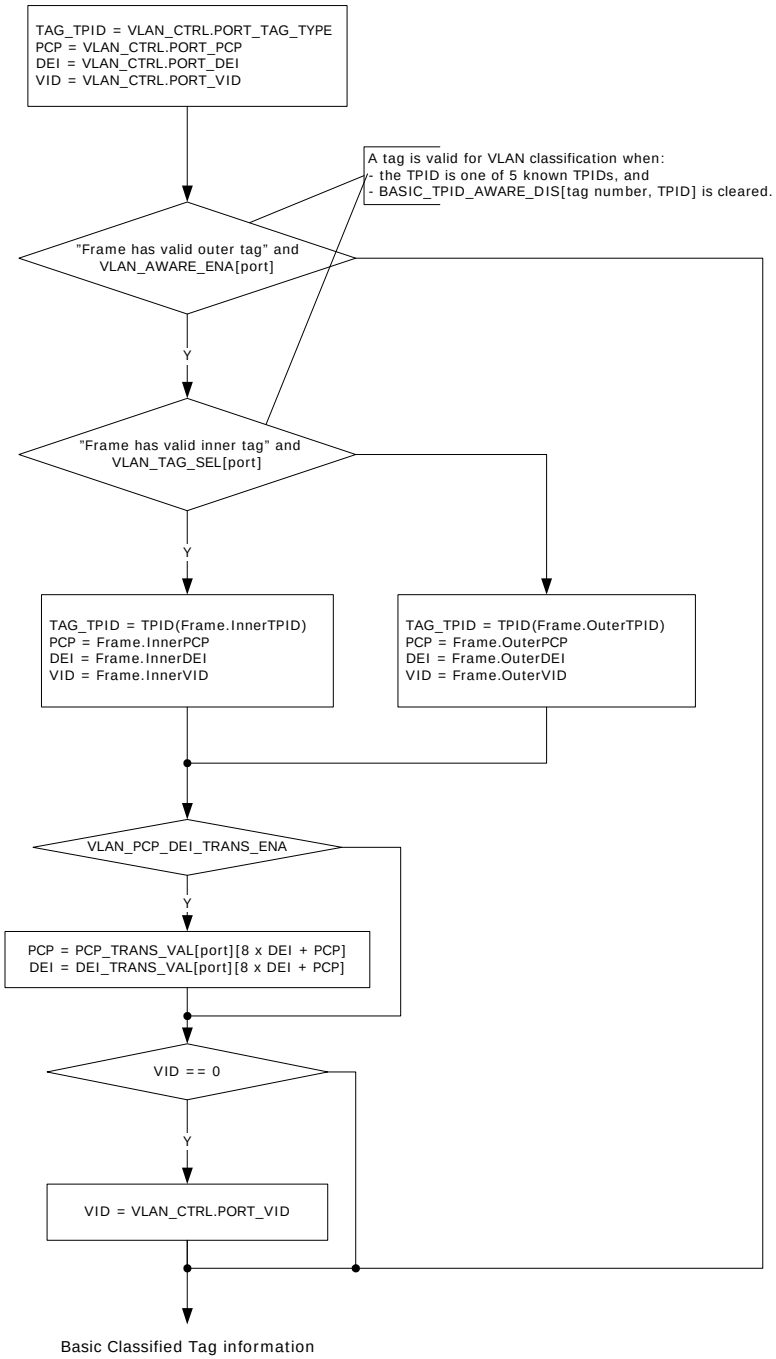
It is configurable, which of the five recognized TPIDs are valid for the VLAN classification (VLAN_TPID_CTRL.BASIC_TPID_AWARE_DIS). Only VLAN tags with valid TPIDs are used by the VLAN classification. The parsing of VLAN tags stops when a VLAN tag with an invalid TPID is seen.

The VLAN tag information is either retrieved from a tag in the incoming frame or from default port-based configuration. The port-based VLAN tag information is configured in ANA_CL:PORT:VLAN_CTRL.

For double tagged frames (at least two VLAN tags with valid TPIDs), there is an option to use the inner tag instead of the outer tag (VLAN_CTRL.VLAN_TAG_SEL). Note that if the frame has more than two valid VLAN tags, the inner tag refers to the tag immediately after the outer tag.

The following illustration shows the flow chart for basic VLAN classification.

Figure 34 • **Basic VLAN Classification Flow Chart**



In addition to the classification shown, the port decides the number of VLAN tags to pop at egress (VLAN_CTRL.VLAN_POP_CNT). If the configured number of tags to pop is greater than the actual number of valid tags in the frame, the number is reduced to the number of actual valid tags in the frame. A maximum of three VLAN tags can be popped. The VLAN pop count is carried in IFH field IFH.TAGGING.POP_CNT.

Finally, the VLAN classification sets IFH field IFH.VSTAX.TAG.WAS_TAGGED if the port is VLAN aware and the frame has one or more valid VLAN tags. WAS_TAGGED is used the rewriter to make decision on whether tag information from existing tags can be used for rewriting.

The basic classified tag information can be overwritten by more intelligent decisions made in the VCAP CLM.

3.14.1.7 Link Aggregation Code Generation

This section provides information about the functions in link aggregation code generation.

The following table lists the registers associated with aggregation code generation.

Table 57 • **Aggregation Code Generation Registers**

Register	Description	Replication
ANA_CL::AGGR_CFG	Configures use of Layer 2 through Layer 4 flow information for link aggregation code generation.	Common

The classifier generates a link aggregation code, which is used in the analyzer when selecting to which port in a link aggregation group a frame is forwarded.

The following contributions to the link aggregation code are configured in the AGGR_CFG register.

- Destination MAC address—use the 48 bits of the DMAC.
- Reversed destination MAC address—use the 48 bits of the DMAC in reverse order (bit 47 becomes bit 0, bit 46 becomes bit 1, and so on).
- Source MAC address—use the lower 48 bits of the SMAC.
- IPv6 flow label—use the 20 bits of the flow label.
- Source and destination IP addresses for IPv4 and IPv6 frames —use all bits of both the SIP and DIP.
- TCP/UDP source and destination ports for IPv4 and IPv6 frames use the 16 bits of both the SPORT and DPORT.
- ISDX—use the 12 bits of the ISDX. The ISDX used is the resulting ISDX after applying the actions from VCAP CLM.
- Random aggregation code—use a 4-bit pseudo-random number.

All contributions that are enabled are 4-bit XOR'ed, and the resulting code is used by the Layer 2 forwarding function (ANA_AC) to select 1 out of 16 link aggregation port masks. The 16 link aggregation port masks are configured in ANA_AC:AGGR. For more information see Aggregation Group Port Selection, page 196.

If AGGR_CFG.AGGR_USE_VSTAX_AC_ENA is enabled and the frame has a VStaX header, all other contributions to link aggregation code calculation are ignored, and the AC field of the VStaX-header is used directly. Otherwise, link aggregation code generation operates as previously described.

Note that AGGR_CFG.AGGR_DMAC_REVERSED_ENA and AGGR_CFG.AGGR_DMAC_ENA are independent. If both bits are enabled, the frame's DMAC contributes both normally and reversed in the link aggregation code calculation.

3.14.1.8 CPU Forwarding Determination

The following table lists the registers associated with CPU forwarding in the basic classifier.

Table 58 • **CPU Forwarding Registers**

Register	Description	Replication
ANA_CL:PORT:CAPTURE_CFG	Enables CPU forwarding for various frame types. Configures valid TPIDs for CPU forwarding.	Per port
ANA_CL:PORT:CAPTURE_BPDU_CFG	Enables CPU forwarding per BPDU address	Per port
ANA_CL:PORT:CAPTURE_GXRP_CFG	Enables CPU forwarding per GxRP address	Per port
ANA_CL:PORT:CAPTURE_Y1731_AG_CFG	Enables CPU forwarding per Y.1731/IEEE802.1ag address	Per port
ANA_CL::CPU_PROTO_QU_CFG	CPU extraction queues for various frame types	None

Table 58 • **CPU Forwarding Registers *(continued)***

Register	Description	Replication
ANA_CL::CPU_8021_QU_CFG	CPU extraction queues for BPDU, GARP, and Y.1731/IEEE802.1ag addresses	None
ANA_CL::VRAP_CFG	VLAN configuration of VRAP filter	None
ANA_CL::VRAP_HDR_DATA	Data match against VRAP header	None
ANA_CL::VRAP_HDR_MASK	Mask used to don't care bits in the VRAP header	None

The basic classifier has support for determining whether certain frames must be forwarded to the CPU extraction queues. Other parts of the device can also determine CPU forwarding, for example, the VCAPs or the VOEs. All events leading to CPU forwarding are OR'ed together, and the final CPU extraction queue mask, which is available to the user, contains the sum of all events leading to CPU extraction.

Upon CPU forwarding by the basic classifier, the frame type and configuration determine whether the frame is redirected or copied to the CPU. Any frame type or event causing a redirection to the CPU causes all front ports to be removed from the forwarding decision—only the CPU receives the frame. When copying a frame to the CPU, the normal forwarding of the frame to front ports is unaffected.

The following table lists the standard frame types recognized by the basic classifier, with respect to CPU forwarding.

Table 59 • **Frame Type Definitions for CPU Forwarding**

Frame	Condition	Copy/Redirect
BPDU Reserved Addresses (IEEE 802.1D 7.12.6)	DMAC = 0x0180C2000000 to 0x0180C20000F	Redirect/Copy/Discard
GARP Application Addresses (IEEE 802.1D 12.5)	DMAC = 0x0180C2000020 to 0x0180C200002F	Redirect/Copy/Discard
Y.1731/IEEE802.1ag Addresses	DMAC = 0x0180C2000030 to 0x0180C200003F	Redirect/Copy/Discard
IGMP	DMAC = 0x01005E000000 to 0x01005E7FFFFF EtherType = IPv4 IP Protocol = IGMP	Redirect
MLD	DMAC = 0x333300000000 to 0x3333FFFFFFFF EtherType = IPv6 IPv6 Next Header = 0 (Hop-by-hop options header) Hop-by-hop options header with the first option being a Router Alert option with the MLD message (Option Type = 5, Opt Data Len = 2, Option Data = 0).	Redirect
Hop-by-hop	EtherType = IPv6 IPv6 Next Header = 0 (Hop-by-hop options header)	Redirect
ICMPv6	EtherType = IPv6 IPv6 Next Header = 58 (ICMPv6)	Redirect
IPv4 Multicast Ctrl	DMAC = 0x01005E000000 to 0x01005E7FFFFF EtherType = IPv4 IP protocol is not IGMP IPv4 DIP inside 224.0.0.x	Copy
IPv6 Multicast Ctrl	DMAC = 0x333300000000 to 0x3333FFFFFFFF EtherType = IPv6 IPv6 header is not hop-by-hop or ICMPv6 IPv6 DIP inside FF02::/16	Copy

CPU forwarding of frame types BPDU and GARP can also be achieved in a more flexible way using an L2CP profile in the IPT table. For more information, see Layer 2 Control Protocol Processing, page 128.

Each frame type has a corresponding CPU extraction queue. Note that hop-by-hop and ICMPv6 frames share the same CPU extraction queue.

Prior to deciding whether to CPU forward a frame, the frame's VLAN tagging must comply with a configurable filter (ANA_CL::CAPTURE_CFG.CAPTURE_TPID_AWARE_DIS). For all frame types listed in Table 59, page 113, only untagged frames or VLAN-tagged frames where the outer VLAN tag's TPID is not disabled are considered.

The basic classifier can recognize VRAP frames and redirect them to the CPU. This is a proprietary frame format, which is used for reading and writing switch configuration registers through Ethernet frames. For more information, see VRAP Engine, page 375.

The VRAP filter in the classifier performs three checks in order to determine whether a frame is a VRAP frame:

1. VLAN check. The filter can be either VLAN unaware or VLAN aware (ANA_CL::VRAP_CFG.VRAP_VLAN_AWARE_ENA). If VLAN unaware, VRAP frames must be untagged. If VLAN aware, VRAP frames must be VLAN tagged and the frame's VID must match a configured value (ANA_CL::VRAP_CFG.VRAP_VID). Double VLAN tagged frames always fail this check.
2. EtherType and EPID check. The EtherType must be 0x8880 and the EPID (bytes 0 and 1 after the EtherType) must be 0x0004.
3. VRAP header check. The VRAP header (bytes 0, 1, 2, and 3 after the EPID) must match a 32-bit configured value (ANA_CL::VRAP_HDR_DATA) where any bits can be don't cared by a mask (ANA_CL::VRAP_HDR_MASK).

If all three checks are fulfilled, frames are redirected to CPU extraction queue ANA_CL::CPU_PROTO_QU_CFG.CPU_VRAP_QU. The VRAP filter is enabled in ANA_CL:PORT:CAPTURE_CFG.CPU_VRAP_REDIR_ENA.

3.14.1.9 Logical Port Mapping

The basic classifier works on the physical port number given by the interface where the frame was received. Other device blocks work on a logical port number that defines the link aggregation instance rather than the physical port. The logical port number is defined in ANA_CL:PORT:PORT_ID_CFG.LPORT_NUM. It is, for instance, used for learning and forwarding in ANA_L2.

3.14.1.10 Port VOE Tag Handling

The following table lists the registers associated with port VOE tag handling in the basic classifier.

Table 60 • **Port VOE Tag Handling Registers Overview**

Register	Description	Replication
ANA_CL:PORT:VLAN_CTRL. PORT_VOE_TPID_AWARE_DIS	Configures which TPIDs are invalid for the port VOE.	Per port
ANA_CL:PORT:VLAN_CTRL. PORT_VOE_DEFAULT_PCP	Default PCP used by the port VOE for untagged or tagged frames with invalid TPID.	Per port
ANA_CL:PORT:VLAN_CTRL. PORT_VOE_DEFAULT_DEI	Default DEI used by the port VOE for untagged or tagged frames with invalid TPID.	Per port

The Rx loss measurement counters in the port VOE use the frame's DEI and PCP. For tagged frames, the DEI and PCP from the frame's outer VLAN tag are used if the VLAN tag's TPID is valid. Invalid TPIDs are configured in VLAN_CTRL.PORT_VOE_TPID_AWARE_DIS. If the VLAN tag's TPID is invalid or the frame is untagged, then port-based default values are used (VLAN_CTRL.PORT_VOE_DEFAULT_DEI and VLAN_CTRL.PORT_VOE_DEFAULT_PCP).

3.14.2 VCAP CLM Processing

Each frame is matched six times against entries in the VCAP CLM. The matching is done in serial implying results from one match can influence the keys used in the next match.

The VCAP CLM returns an action for each enabled VCAP CLM lookup. If the lookup matches an entry in VCAP CLM, the associated action is returned. If an entry is not matched, a per-port default action is returned. There is no difference between an action from a match and a default action.

3.14.2.1 VCAP CLM Port Configuration and Key Selection

This section provides information about special port configurations that control the key generation for VCAP CLM.

The following table lists the registers associated with port configuration for VCAP CLM.

Table 61 • **Port Configuration of VCAP CLM**

Register	Description	Replication
ANA_CL:PORT:ADV_CL_CFG	Configuration of the key selection for the VCAP CLM. Enables VCAP CLM lookups.	Per port per lookup
ANA_CL::CLM_MISC_CTRL	Force lookup in VCAP CLM for looped frames or frames where processing is terminated by a pipeline point.	None

VCAP CLM is looked up if the lookup is enabled by the port configuration (ADV_CL_CFG.LOOKUP_ENA) or if the previous VCAP CLM lookup has returned a key type for the next lookup through VCAP CLM action NXT_KEY_TYPE. However, if the previous lookup returns NXT_KEY_TYPE = CLMS_DONE, then no further lookups are done, even if the port has enabled the lookup.

Prior to each VCAP CLM lookup, the type of key to be used in the lookup is selected. A number of parameters control the key type selection:

- Configuration parameters (ADV_CL_CFG).
  For each (port, VCAP CLM lookup) a key type can be specified for different frame types. Although any key type can be specified for a given frame type, not all key types are in reality applicable to a given frame type. The applicable key types are listed in the ANA_CL:PORT:ADV_CL_CFG register description. The following frame types are supported:
  - IPv6 frames (EtherType 0x86DD and IP version 6)
  - IPv4 frames (EtherType 0x0800 and IP version 4)
  - MPLS unicast frames (EtherType 0x8847)
  - MPLS multicast frames (EtherType 0x8848)
  - MPLS label stack processing (frame type after MPLS link layer is processed)
  - Other frames (non-MPLS, non-IP)
- The type of the current protocol layer. For each VCAP CLM lookup, a frame pointer can optionally be moved to the next protocol layer, and the type of the next protocol layer can be specified. This influences key type selection for the next VCAP CLM lookup. The variables frame_ptr and frame_type holds information about the current protocol layer.

Key type specified in action in previous VCAP CLM lookup.
The action of a VCAP CLM lookup can optionally specify a specific key type to be used in the following VCAP CLM lookup.

The full algorithm used for selecting VCAP CLM key type is shown in the following pseudo code.

```
// ================================================================
// ANA_CL: VCAP CLM Key Type Selection
//
// Determine which key type to use in VCAP CLM lookup.
// The algorithm is run prior to each VCAP CLM lookup.
// ----------------------------------------------------------------
// Frame position (byte pointer) for use as base for next VCAP CLM key
```

```
// Updated by UpdateFramePtrAndType() upon VCAP CLM lookup
int frame_ptr = 0;

// Frame type at frame_ptr position.
// Updated by UpdateFramePtrAndType() upon VCAP CLM lookup
//
// Supported values:
//   ETH:  frame_ptr points to DMAC
//   CW:   frame_ptr points to start of IP header or CW/ACH of MPLS frame
//   MPLS: frame_ptr points to an MPLS label.
frame_type_t frame_type = ETH;

int ClmKeyType(
               // VCAP CLM index. 0=First VCAP CLM lookup, 5=Last lookup
               clm_idx,

               // Action from previous VCAP CLM lookup (if clm_idx > 0)
               //
               // clm_action.vld is only set if VCAP CLM lookup
               // was enabled.
               clm_action,
               )
{
  int        key_type = 0;
  port_cfg_t port_clm_cfg;
  port_clm_cfg = csr.port[port_num].adv_cl_cfg[clm_idx];

  if (clm_action.vld && clm_action.nxt_key_type != 0) {
    // Previous VCAP CLM lookup specifies next key type
    return clm_action.nxt_key_type;
  }

  // Determine key type based on port parameters and frame_ptr/frame_type
  switch (frame_type) {
  case ETH:
    // Choose key type based on EtherType
    // The EtherType() function skips any VLAN tags. A maximum of 3 VLAN
    // tags can be skipped.
    switch (EtherType(frame_ptr)) {
    case ETYPE_IP4: // 0x0800
      key_type = port_clm_cfg.ip4_clm_key_sel;
      break;

    case ETYPE_IP6: // 0x86DD
      key_type = port_clm_cfg.ip6_clm_key_sel;
      break;

    case ETYPE_MPLS_UC: // 0x8847
      key_type = port_clm_cfg.mpls_uc_clm_key_sel;
      break;
    case ETYPE_MPLS_MC: // 0x8848
      key_type = port_clm_cfg.mpls_mc_clm_key_sel;
      break;
    }
    if (key_type = 0) {
      key_type = port_clm_cfg.etype_clm_key_sel;
    }
    break;
```

```
  case CW:
    switch (IPVersion(frame_ptr)) {
    case 4 : key_type = port_clm_cfg.ip4_clm_key_sel;   break;
    case 6 : key_type = port_clm_cfg.ip6_clm_key_sel;   break;
    default: key_type = port_clm_cfg.etype_clm_key_sel; break;
    }
    break;
  case MPLS:
    key_type = port_clm_cfg.mlbs_clm_key_sel;
    break;
  }
  return key_type;
} // ClmKeyType

// ------------------------------------------------------------------
// ANA_CL: VCAP CLM Key Type Selection
// ==================================================================
```

By default, frames looped by the rewriter or frames already discarded or CPU-redirected, for instance the basic classifier, are not subject to VCAP CLM lookups. Optionally, VCAP CLM lookups can be enforced through configuration ANA_CL::CLM_MISC_CTRL.LBK_CLM_FORCE_ENA for looped frames and through ANA_CL::CLM_MISC_CTRL.IGR_PORT_CLM_FORCE_ENA for frames already discarded or CPU-redirected. When forcing a VCAP CLM lookup, the VCAP CLM key is chosen from the configuration in ANA_CL::CLM_MISC_CTRL.FORCED_KEY_SEL. Only selected keys are available.

3.14.2.2 VCAP CLM MPLS Key Field Extraction

When the frame_type is either MPLS or ETH with an MPLS EtherType, then VCAP CLM key types SGL_MLBS, DBL_MLBS and TRI_MLBS can be used. For more information, see the pseudo code in VCAP CLM Port Configuration and Key Selection, page 115.

These keys contain information about 1, 2, or 3 label stack entries (LSE). These LSEs are generally the LSEs at the top of the label stack or at the position pointed to by frame_ptr, depending on frame_type.

However, for each of the reserved MPLS label values (0-15), it is possible to have the label value placed in a separate field in the MPLS VCAP CLM keys. This is termed “reserved label skipping”. Skipping reserved labels enables that both OAM traffic as well as service traffic utilize the same VCAP CLM entry, thus increasing fate sharing and reducing number of VCAP CLM entries required. In such case a reserved label value is placed in VCAP CLM key field RSV_LBL_VAL and the position of the LSE is placed in VCAP CLM key field RSV_LBL_POS.

The following parameters controls if reserved label values are skipped.

Table 62 • **Reserved Label Skipping Configuration**

Register	Description	Replication
ANA_CL::MPLS_RSV_LBL_CFG	Configures skip of label per reserved label during label extract.	Per reserved label
ANA_CL::MPLS_MISC_CFG	Enable skipping of reserved label during label extract.	Per VCAP CLM lookup
ANA_CL::OAM_CFG	Configuration of Reserved Label used for PW VCCV2 OAM channel.	Common
ANA_CL:MPLS_PROFILE	Configures further processing of non-skipped reserved labels and IP control frames.	Per reserved label, per IP protocol (IPv4 and IPv6)
ANA_CL::MPLS_LM_CFG	Configures MPLS OAM Loss Measurement handling.	Common

The following pseudo code shows the algorithm used for VCAP CLM MPLS key field extraction. The algorithm is run prior to each VCAP CLM lookup.

```
//=============================================================================
// ANA_CL: MPLS VCAP CLM Key Field Extraction
//
// This algorithm is used to extract VCAP CLM key fields for keys of types
// SGL_MLBS, DBL_MLBS and TRI_MLBS.
// Only TRI_MLBS uses all the extracted fields.
// The algorithm is run prior to each VCAP CLM lookup.
// -----------------------------------------------------------------------

clm_key_fields_mpls_t ClmKeyFieldsMpls(clm_idx) {
  // Result variable
  clm_key_fields_mpls_t clm_key_fields_mpls;

  // Initialize all fields to 0 (the default if label not present/found)
  for (i = 0; i < 3; i++) {
    clm_key_fields_mpls.lbl[i]        = 0; // LBL0 - LBL2
    clm_key_fields_mpls.tc[i]         = 0; // TC0 - TC2
    clm_key_fields_mpls.sbit[i]       = 0; // SBIT0 - SBIT2
    clm_key_fields_mpls.ttl_expiry[i] = 0; // TTL0_EXPIRY - TTL2_EXPIRY
  }
  clm_key_fields_mpls.rsv_lbl_val = 0; // Used by key TRI_MLBS
  clm_key_fields_mpls.rsv_lbl_pos = 0; // Used by keys TRI_MLBS, DBL_MLBS

  // Get pointer to first MPLS LSE based on frame_ptr and frame_type as
  // updated by UpdateFramePtrAndType()
  mpls_next_lse_ptr = GetFirstLsePtr(frame_ptr, frame_type);
  clm_mpls_key_idx = 0;
  mpls_lse_idx     = 0;
  prev_lbl_rsv     = false; // Previous label was reserved

  // Extract up to 3 LSEs, if present
  while (clm_mpls_key_idx < 3) {
    cur_lbl_rsv = false; // Current label is reserved

    // Get next MPLS LSE
    lse = GetLse(mpls_next_lse_ptr);
    mpls_next_lse_ptr += 4; // Move to next (potential) LSE position

    // "Skipping" of (selected) reserved labels enabled
    if (lse.label < 16 &&
        csr.common.mpls_misc_cfg.clm.rsvd_lbl_skip_ena[clm_idx] &&
        csr.common.mpls_rsv_lbl_cfg[lse.label].rsvd_lbl_skip_ena) {
      // Reserved label found
      if (clm_key_fields_mpls.rsv_lbl_pos == 0) {
        // First reserved label found
        cur_lbl_rsv = true;
        clm_key_fields_mpls.rsv_lbl_pos = mpls_lse_idx + 1;
        clm_key_fields_mpls.rsv_lbl_val = lse.label;

        if (csr.common.oam_cfg.vccv2_ena &&
            csr.common.oam_cfg.vccv2_label == lse.label) {
          // Get one additional LSE below this reserved label
// (if present)
        } else {
          // Dig no deeper
          return clm_key_fields_mpls;
        }
      }
    } else {
```

```
      clm_key_fields_mpls.lbl[clm_mpls_key_idx]        = lse.label;
      clm_key_fields_mpls.tc[clm_mpls_key_idx]         = lse.tc;
      clm_key_fields_mpls.sbit[clm_mpls_key_idx]       = lse.s;
      clm_key_fields_mpls.ttl_expiry[clm_mpls_key_idx] = (lse.ttl <= 1);
      clm_mpls_key_idx++;
    }

    if (prev_lbl_rsv) {
      // Never look more than one LSE below a reserved label
      return clm_key_fields_mpls;
    }
    prev_lbl_rsv = cur_lbl_rsv;
    mpls_lse_idx++;

    if (lse.s) {
      // Bottom of stack found
      return clm_key_fields_mpls;
    }
  } // while ...
} // ClmKeyFieldsMpls
// ------------------------------------------------------------------------
// ANA_CL: MPLS VCAP CLM Key Field Extraction
// ========================================================================
```

3.14.2.3 Frame Pointer and Frame Type Update

After each VCAP CLM lookup, the information in the associated action is used to update frame_ptr and frame_type. This in turn influences the key type and key field values used for the next VCAP CLM lookup.

The following fields in the VCAP CLM action can be used to move to the next protocol layer:

- NXT_OFFSET_FROM_TYPE.
  Move frame pointer past current protocol layer (Ethernet and/or IP). The actual number of bytes the frame pointer is moved may vary depending on content of the current protocol layer, such as the number of VLAN tags or IPv4 options.
- NXT_NORM_W16_OFFSET.
  Move frame pointer a number of 16 bit words. If both NXT_OFFSET_FROM_TYPE and NXT_NORM_W16_OFFSET are specified, then frame pointer is first moved based on NXT_OFFSET_FROM_TYPE and afterwards moved further into the frame based on NXT_NORM_W16_OFFSET.
- NXT_TYPE_AFTER_OFFSET.
  Specify protocol layer at new frame pointer position.

For more information about VCAP CLM action fields, see Table 51, page 91.

Each VCAP CLM lookup can at most move the frame pointer 66 bytes. The following pseudo code shows the algorithm used to update frame_ptr and frame_type upon each VCPA_CLM lookup.

```
// ========================================================================
// ANA_CL: VCAP CLM Frame Pointer and Frame Type Update
//
// Update frame_ptr and frame_type upon VCAP CLM lookup.
// Prior to first VCAP CLM lookup frame_ptr is initialized to 0 and
// frame_type is set to ETH.
// The algorithm is run after each VCAP CLM lookup.
// ------------------------------------------------------------------------

// Function for updating
//   frame_ptr
//   frame_type
// based on action from VCAP CLM lookup
```

```
void UpdateFramePtrAndType(
                           // VCAP CLM index. 0=First VCAP CLM lookup,
                           // 5=Last VCAP CLM lookup
                           clm_idx,

                           // Action from VCAP CLM lookup
                           // clm_action.vld is only set if
                           // VCAP CLM lookup was enabled.
                           clm_action,

                           // From ClmKeyFieldsMpls()
                           clm_key_fields_mpls,
                           )
{
  int frame_ptr_current  = frame_ptr;
  int frame_type_current = frame_type;

  if (!clm_action.vld) return;

  if (clm_action.nxt_offset_from_type > 0) {
    // Move frame_ptr based on frame type
    switch (clm_action.nxt_offset_from_type) {
    case SKIP_ETH:
      if (frame_type_current == ETH) {
        // Move frame_ptr past DMAC, SMAC, 0-3 VLAN tags and EtherType
        SkipLinkLayer(&frame_ptr);
      }
      break;

    case SKIP_IP_ETH:
      switch (frame_type_current) {
      case ETH:
        // Move frame_ptr past DMAC, SMAC, 0-3 VLAN tags and EtherType
        if (EtherType(frame_ptr) == ETYPE_IP4) || // 0x0800
           (EtherType(frame_ptr) == ETYPE_IP6) {  // 0x86DD
          SkipLinkLayer(&frame_ptr);

          // Move frame_ptr past IPv4 header (including options) or
          // IPv6 header
          SkipIPHeader(&frame_ptr);
        }
        break;

      case CW:
        switch (IPVersion(frame_ptr)) {
        case 4:
        case 6:
          // Move frame_ptr past IPv4 header (including options) or
          // IPv6 header
          SkipIPHeader(&frame_ptr);
          break;
        }
        break;
      }
    }

    frame_type = clm_action.nxt_type_after_offset;
  } // clm_action.nxt_offset_from_type > 0
```

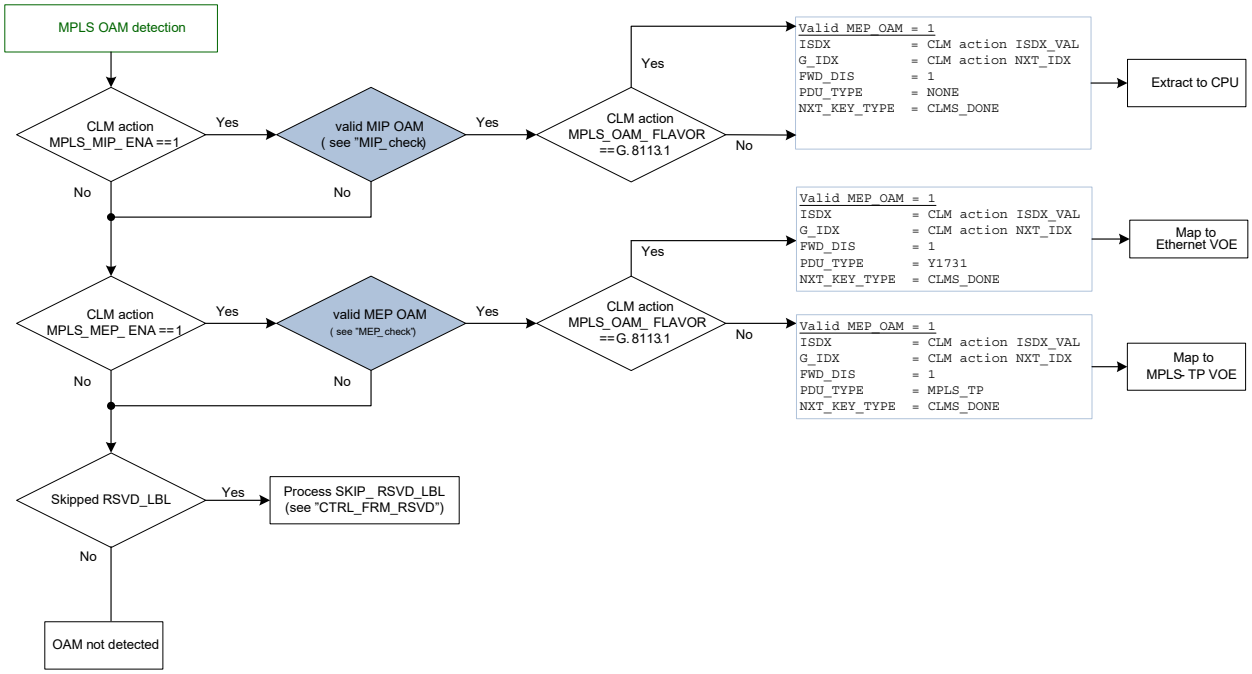
```
  if (clm_action.nxt_norm_w16_offset > 0) {
    // Move frame_ptr a specific number of bytes
    frame_ptr  = frame_ptr + 2*clm_action.nxt_norm_w16_offset;
    frame_type = clm_action.nxt_type_after_offset;
  }

  if (frame_type_current == DATA &&
      clm_action.nxt_key_type != 0) {
    // Allow frame_type change, regardsless of whether frame_ptr
    // has been moved
    frame_type = clm_action.nxt_type_after_offset;
  }

  // PW termination of MPLS frame
  if (clm_action.fwd_type == TERMINATE_PW) {
    if IsOam(frame_ptr, csr) && (clm_key_fields_mpls.rsv_lbl_pos) {
      // Reserved label used to identify OAM
      // Move frame_ptr if reserved MPLS label was skipped by
      // ClmKeyFieldsMpls() in key generation prior to VCAP CLM lookup
      frame_ptr += 4;
    } elsif (frame_type == CW && !IsOam(frame_ptr, csr)) {
      // PW termination of MPLS frame with CW => Skip CW and
      // change frame_type
      frame_ptr += 4;
      frame_type = ETH;
    }
  }

  // OAM LSP
  if (clm_action.fwd_type == POP_LBL && IsOam(frame_ptr, csr) &&
      (clm_key_fields_mpls.rsv_lbl_pos)) {
    // Reserved label used to identify OAM
    // Move frame_ptr if reserved MPLS label was skipped by
    // ClmKeyFieldsMpls() in key generation prior to VCAP CLM lookup
    frame_ptr += 4;
  }

  if (frame_ptr - frame_ptr_current > 66) {
    // Cannot skip >66 bytes per VCAP CLM.
    // Set sticky bit and don't skip anything
    csr.adv_cl_max_w16_offset_fail_sticky = 1;
    frame_ptr  = frame_ptr_current;
    frame_type = frame_type_current;
  }

  if (frame_ptr > 126 ) {
    // Cannot skip >126 bytes total.
    // Set sticky bit and discard frame
    csr.adv_cl_nxt_offset_too_big_sticky = 1;
    frame.abort  = true;
  }
} // UpdateFramePtrAndType

// ------------------------------------------------------------------------
// ANA_CL: VCAP CLM Frame Pointer and Frame Type Update
// ========================================================================
```

3.14.2.4 Detecting MPLS OAM PDU

The device supports software MIP and hardware MEP handling of MPLS OAM frames carried over LSPs and pseudowires (PWs). Frames detected as MEP OAM can be processed by the VOP_MPLS block, whereas frames detected as MIP OAM are sent to the CPU for further software processing.

MPLS OAM detection is shown in the following illustration.

Figure 35 • **MPLS OAM Detection**



MPLS OAM for pseudowires is called Virtual Circuit Connection Verification (VCCV), as specified in RFC 5085 and RFC 5586. VCCV uses an exception handling mechanism to identify the OAM channel VCCV. The hardware of a particular PW OAM is configured to support at most one of the VCCV configuration options when terminating the PW through VCAP CLM action FWD_TYPE = 1:

- VCCV1 - Associated channel (ACH), first nibble = 0x1:
  VCAP CLM action: MPLS_OAM_TYPE = 1.
- VCCV2 - Router alert label (RAL):
  VCAP CLM action: MPLS_OAM_TYPE = 2.
- VCCV3 - PW with TTL = 1:
  VCAP CLM action: MPLS_OAM_TYPE = 3.
- VCCV4 - GAL followed by ACH:
  VCAP CLM action: MPLS_OAM_TYPE = 4.

Hardware handling of LSP OAM is configured by popping the LSP label and by setting the MPLS OAM type to LSP OAM. This is configured through VCAP CLM actions FWD_TYPE = 3 (pop LSP) and MPLS_OAM_TYPE = 5 (LSP OAM).

Hardware handling of Segment OAM is configured through VCAP CLM actions FWD_TYPE = 4 (CTRL_PDU) and MPLS_OAM_TYPE = 2 (VCCV2).

The device supports OAM detection of both G.8113.1 and G.8113.2 OAM configured through VCAP CLM action MPLS_OAM_FLAVOR. This setting determines whether the G-ACH/ACH selected by MPLS_OAM_TYPE is checked for Channel Type = 0x8902. This configuration applies only to MEP OAM detection. OAM MIP detection is extracted to software based on the configured Control Channel Type, regardless of the channel type, because all OAM MIP processing is done in software.

3.14.2.5 Handling of Reserved Labels

The device supports processing reserved labels that are not used for detecting OAM.

Skipped reserved labels within a number of valid labels for a given CLM entry (CLM action NUM_VLD_LABELS) are based on configuration (ANA_CL:MPLS_PROFILE:PROFILE_CFG.FWD_SEL) either redirected to configurable CPU queue (ANA_CL:MPLS_PROFILE:PROFILE_CFG.CPU_QU) or discarded.

Non-skipped reserved labels that are exposed when terminated PW (VCAP CLM action FWD_TYPE = 1) or when popping an LSP (VCAP CLM action FWD_TYPE = 3) are handled based on a profile configuration (ANA_CL:MPLS_PROFILE:PROFILE_CFG). There is a profile for each reserved label, and the following controls are available per profile:

- Forward handling (ANA_CL:MPLS_PROFILE:PROFILE_CFG.FWD_SEL and ANA_CL:MPLS_PROFILE:PROFILE_CFG.CPU_QU)
- Next VCAP CLM key lookup control (ANA_CL:MPLS_PROFILE:PROFILE_CFG.NXT_KEY_TYPE)
- Frame normalization and pointer movements (ANA_CL:MPLS_PROFILE:PROFILE_CFG.NORMALIZE_DIS, ANA_CL:MPLS_PROFILE:PROFILE_CFG.NXT_NORM_W16_OFFSET and ANA_CL:MPLS_PROFILE:PROFILE_CFG.NXT_TYPE_AFTER_OFFSET)
- Controls if profile traffic should be part of OAM loss measurement (ANA_CL:MPLS_PROFILE:PROFILE_CFG.LM_CNT_DIS)
- VCAP IS2 custom key lookup (ANA_CL:MPLS_PROFILE:PROFILE_CFG.CUSTOM_ACE_ENA)

For example, MPLS encapsulated IPv4 Frames received with an explicit null label can be configured to perform a NORMAL_5TUPLE_IP4 lookup for the next VCAP CLM lookup:

- Allow normal forwarding: ANA_CL:MPLS_PROFILE:PROFILE_CFG.FWD_SEL = 0
- Use key type NORMAL_5TUPLE_IP4 in the next VCAP CLM lookup: ANA_CL:MPLS_PROFILE:PROFILE_CFG.NXT_KEY_TYPE = 10
- Frame normalization and pointer movements: ANA_CL:MPLS_PROFILE:PROFILE_CFG.NORMALIZE_DIS = 0 ANA_CL:MPLS_PROFILE:PROFILE_CFG.NXT_NORM_W16_OFFSET = 2 ANA_CL:MPLS_PROFILE:PROFILE_CFG.NXT_TYPE_AFTER_OFFSET = 1
- Part of OAM loss measurements: ANA_CL:MPLS_PROFILE:PROFILE_CFG.LM_CNT_DIS = 0
- No explicit VCAP IS2 lookup: ANA_CL:MPLS_PROFILE:PROFILE_CFG.CUSTOM_ACE_ENA = 0

3.14.2.6 Handling of IP Control Frames

The device supports handling of IP traffic that are exposed when popping a LSP (VCAP CLM action FWD_TYPE = 3) based on IP profile configuration (ANA_CL:MPLS_PROFILE:PROFILE_CFG index 16 for IPv4 and 17 for IPv6). These profiles allow the same controls as available for reserved labels.

VCAP CLM action MPLS_IP_CTRL_ENA controls if IP control protocols are handled through the IP profiles (ANA_CL:MPLS_PROFILE:PROFILE_CFG) or treated as unexpected IP frames and redirected to the CPU (ANA_CL:COMMON:MPLS_CFG.CPU_MPLS_IP_TRAFFIC_QU).

It is configurable whether or not unexpected IP frames are part of OAM loss measurement (ANA_CL:COMMON:MPLS_LM_CFG.MPLS_IP_ERR_LM_CNT_DIS).

3.14.2.7 Detecting SAM Sequence

The device supports the detection of SAM sequence frames using VCAP CLM actions to enable SAM sequence updates in the VOP. This is configured through VCAP CLM actions FWD_TYPE = 4 (CTRL_PDU) and MPLS_OAM_TYPE = 7. This allows the VOP to identify the flow as being a SAM sequence flow and makes the appropriate frame updates. For more information, see Non-OAM Sequence Numbering, page 265.

3.14.2.8 Removing Protocol Layers

The analyzer classifier can instruct the rewriter to remove protocol layers from the frame, that is, to remove a number of bytes from the start of the frame.

This is achieved by setting the NXT_NORMALIZE action field in the VCAP CLM action. When NXT_NORMALIZE is set in a VCAP CLM action, then the frame pointer for the next VCAP CLM lookup (if any) is calculated, and the rewriter is then instructed to remove all bytes up to, but not including, the position pointed to by the frame pointer.

If multiple VCAP CLM actions have the NXT_NORMALIZE bit set, then the last such action controls the number of bytes removed by the rewriter.

Note that the rewriter can skip a maximum of 42 bytes, starting at the DMAC of the frame. This corresponds to an Ethernet header with three VLAN tags and an MPLS stack with three LSEs and a control word (CW).

3.14.2.9 Miscellaneous VCAP CLM Key Configuration

The analyzer classifier can control some specific fields when generating the VCAP CLM keys. The following table lists the registers that control the parameters.

Table 63 • **Miscellaneous VCAP CLM Key Configuration**

Register	Description	Replication
ANA_CL:PORT:ADV_CL_CFG	Configures L3_DSCP and TCI0 source.	Per port per VCAP CLM lookup
ANA_CL::CLM_MISC_CTRL	Configures IGR_PORT_MASK_SEL and IGR_PORT_MASK.	None

Each port can control specific fields in the generated keys:

- Selects source of DSCP in L3_DSCP key field (ANA_CL:PORT:ADV_CL_CFG.USE_CL_DSCP_ENA). By default, the DSCP value is taken from the frame. Another option is to use the current classified DSCP value as input to the key. The current classified DSCP value is the result from either the previous VCAM_CLM lookup or the basic classifier if this is the first lookup.
- Selects source of VID, PCP, and DEI in the VID0, PCP0, and DEI0 key fields (ANA_CL:PORT:ADV_CL_CFG.USE_CL_TCI0_ENA). By default, the values are taken from the outer VLAN tag in the frame (or port's VLAN if frame is untagged). Another option is to use the current classified values as input to the key. The current classified values are the result from either the previous VCAM_CLM lookup or the basic classifier if this is the first lookup.

Note that these configurations are only applicable to keys containing the relevant key fields.

VCAP CLM keys NORMAL, NORMAL_7TUPLE, and NORMAL_5TUPLE_IP4 contain the key fields IGR_PORT_MASK_SEL and IGR_PORT_MASK. For frames received on a front port, IGR_PORT_MASK_SEL is set to 0 and the bit corresponding to the frame's physical ingress port number is set in IGR_PORT_MASK. The following lists some settings for frames received on other interfaces and how this can be configured through register CLM_MISC_CTRL:

- Loopback frames:
  By default, IGR_PORT_MASK_SEL=1 and IGR_PORT_MASK has one bit set corresponding to the physical port which the frame was looped on. Optionally use IGR_PORT_MASK_SEL = 3.
- Masqueraded frames:
  By default, IGR_PORT_MASK_SEL=0 and IGR_PORT_MASK has one bit set corresponding to the masquerade port. Optionally, use IGR_PORT_MASK_SEL = 2.
- Virtual device frames:
  By default, IGR_PORT_MASK_SEL = 0 and IGR_PORT_MASK has one bit set corresponding to the original physical ingress port. Optionally, use IGR_PORT_MASK_SEL = 3.
- CPU injected frames:
  By default, IGR_PORT_MASK_SEL=0 and IGR_PORT_MASK has none bits set. Optionally use IGR_PORT_MASK_SEL=3.

For more information about the contents of IGR_PORT_MASK_SEL and IGR_PORT_MASK, see VCAP CLM Keys and Actions, page 70.

3.14.2.10 VCAP CLM Range Checkers

The following table lists the registers associated with configuring VCAP CLM range checkers.

Table 64 • **VCAP CLM Range Checker Configuration Registers Overview**

Register	Description	Replication
ANA_CL::ADV_RNG_CTRL	Configures range checker types	Per range checker
ANA_CL::ADV_RNG_VALUE_CFG	Configures range start and end points	Per range checker

Eight global VCAP CLM range checkers are supported by the NORMAL, NORMAL_7TUPLE, NORMAL_5TUPLE_IP4, and PURE_5TUPLE_IP4 keys. All frames using these keys are compared against the range checkers and a 1-bit range “match/no match” flag is returned for each range checker. The combined eight match/no match flags are used in the L4_RNG key field. So, it is possible to include for example ranges of DSCP values and/or ranges of TCP source port numbers in the VCAP rules.

Note: VCAP IS2 also supports range checkers, however, they are independent of the VCAP CLM range checkers.

The range key is generated for each frame based on extracted frame data and the configuration in ANA_CL::ADV_RNG_CTRL. Each of the eight range checkers can be configured to one of the following range types:

- TCP/UDP destination port range
  Input to the range is the frame’s TCP/UDP destination port number.
  Range is only applicable to TCP/UDP frames.
- TCP/UDP source port range
  Input to the range is the frame’s TCP/UDP source port number.
  Range is only applicable to TCP/UDP frames.
- TCP/UDP source and destination ports range. Range is matched if either source or destination port is within range.
  Input to the range are the frame’s TCP/UDP source and destination port numbers.
  Range is only applicable to TCP/UDP frames.
- VID range
  Input to the range is the classified VID.
  Range is applicable to all frames.
- DSCP range
  Input to the range is the classified DSCP value.
  Range is applicable to IPv4 and IPv6 frames.
- EtherType range
  Input to the range is the frame’s EtherType.
  Range is applicable to all frames.

Range start points and range end points are configured in ANA_CL::ADV_RNG_VALUE_CFG with both the start point and the end point being included in the range. A range matches if the input value to the range (for instance the frame’s TCP/UDP destination port) is within the range defined by the start point and the end point.

3.14.2.11 Cascading VCAP CLM Lookups

Each of the six VCAP CLM lookups can be “cascaded” such that a frame only matches a VCAP CLM entry in lookup n+1 if it also matched a specific entry in VCAP CLM lookup *n*. In other words, VCAP CLM entries in different VCAP CLM lookups are bound together.

The following parameters controls cascading of VCAP CLM entries:

- Key field: G_IDX
  The generic index, G_IDX, is available in most VCAP CLM key types. The value of this field can be specified in the action of the preceding VCAP CLM lookup.
- Action fields: NXT_IDX_CTRL and NXT_IDX
  Control how to calculate G_IDX for next VCAP CLM lookup.

For information about detailed encoding of NXT_IDX_CTRL, see Table 51, page 91.

In addition to cascading VCAP CLM lookups, it is also possible for each VCAP CLM lookup to move to the next protocol layer of the frame, such that key values for lookup *n* + 1 are retrieved from a different offset in the frame than for lookup *n*.

3.14.3 QoS Mapping Table

The following table lists the registers associated with the QoS mapping table.

Table 65 • **VCAP CLM QoS Mapping Table Registers Overview**

Register	Description	Replication
ANA_CL:MAP_TBL:SET_CTRL	Controls which mapping results to use.	256
ANA_CL:MAP_TBL:MAP_ENTRY	Configuration of the QoS mapping values.	2,048

The classifier contains a shared QoS mapping table with 2,048 entries (ANA_CL:MAP_TBL:MAP_ENTRY) that maps incoming QoS information to classified internal QoS information. Inputs to a mapping can be either DEI and PCP from VLAN tags, DSCP value, or MPLS TC bits. Outputs from a mapping are new classified values for COSID, DEI, PCP, DSCP, QoS class, DP level, and TC bits. The table is looked up twice for each frame with two different keys.

The QoS mapping table is laid out with 256 rows with each eight mapping entries. Each specific QoS mapping only uses a subset of the 2,048 entries and it therefore allows for many QoS different mappings at the same time. It can for instance hold 32 unique mappings from DSCP to DEI/PCP or 128 unique translations of DEI/PCP to DEI/PCP.

The two lookups per frame in the QoS mapping table are defined by the VCAP CLM actions MAP_IDX and MAP_KEY. One VCAP CLM match can only define one set MAP_IDX and MAP_KEY so two lookups in the QoS mapping table requires two VCAP CLM matches. Action MAP_IDX defines which one of the 256 rows the QoS mapping starts at and the MAP_KEY defines which parameter from the frame to use as key into the QoS mapping table. The following lists the available keys and how the resulting row and entry number in the mapping table is derived:

- **PCP.** Use PCP from the frame's outer VLAN tag. If the frame is untagged, the port's VLAN tag is used. Row and entry number is derived the following way:
  Row = MAP_IDX, Entry = PCP.

  Number of entries per mapping: 8 entries.
- **DEI and PCP.** Use DEI and PCP from one of the frame's VLAN tags. The MAP_KEY can select between using either outer, middle, or inner tag. If the frame is untagged, the port's VLAN tag is used. Row and entry number is derived the following way:
  Row = (8x DEI + PCP) DIV 8 + MAP_IDX, Entry = (8x DEI + PCP) MOD 8.

  Number of entries per mapping: 16 entries.
- **DSCP (IP frames only).** Use the frame's DSCP value. For non-IP frames, no mapping is done. Row and entry number is derived the following way:
  Row = DSCP DIV 8 + MAP_IDX, Entry = DSCP MOD 8.

  Number of entries per mapping: 64 entries.
- **DSCP (all frames).** Use the frame's DSCP value. For non-IP VLAN tagged frames, use DEI and PCP from the frame's outer VLAN tag. For non-IP untagged frames, use DEI and PCP from port VLAN. Row and entry number is derived the following way:
  IP frames:
  Row = DSCP DIV 8 + MAP_IDX, Entry = DSCP MOD 8.

  Non-IP frames:
  Row = (8x DEI + PCP) DIV 8 + MAP_IDX + 8, Entry = (8x DEI + PCP) MOD 8.

  Number of entries per mapping: 80 entries.
- **TC.** Use the frame's classified TC bits from the extracted MPLS label (label selected by VCAP CLM actions TC_ENA or TC_LABEL). Row and entry number is derived the following way:

Row = MAP_IDX, Entry = TC.

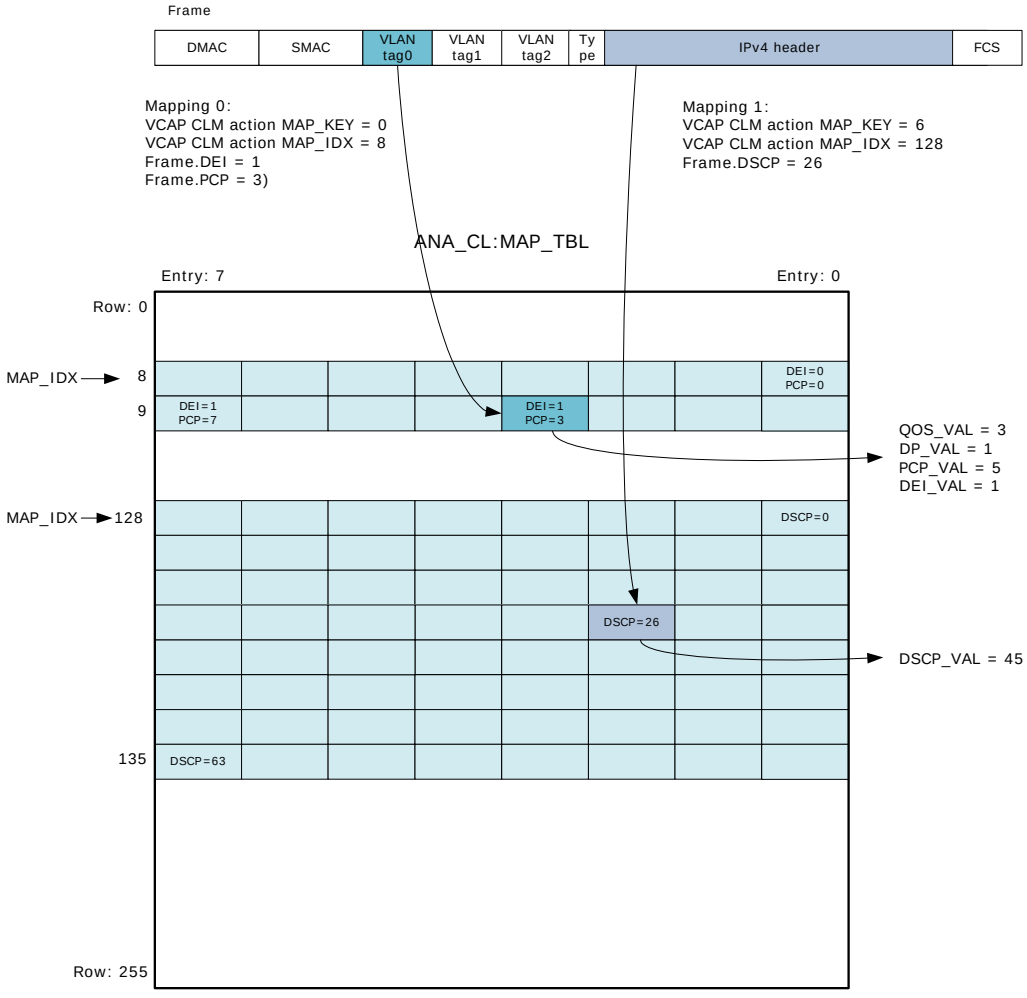
Number of entries per mapping: 8 entries.

QoS mappings can be shared between multiple VCAP CLM entries by defining the same mapping and using the same entries in the mapping table. It is up to the user of the QoS mapping table to make sure that entries are allocated so that they do not overlap unintentionally.

Each entry in the QoS mapping table (ANA_CL:MAP_TBL:MAP_ENTRY) contains a full set of new QoS values (COSID, DEI, PCP, QoS class, DP level, DSCP, TC) to be used as new classified values instead of the current classified values from basic classification and VCAP CLM. Each of the QoS values is gated by an enable bit (ANA_CL:MAP_TBL:SET_CTRL) so that a mapping can for example define a complete new set of QoS values by enabling all values or it can override selected values only. Note that ANA_CL:MAP_TBL:SET_CTRL must be configured for each used row in the mapping table.

The following illustration shows an example of a frame for which a DSCP and a DEI/PCP mapping are defined.

Figure 36 • **Example of QoS Mappings**



3.14.4 Ingress Protection Table (IPT)

The analyzer classifier contains an ingress protection table (IPT) that enables per-ISDX configuration related to protection, MIP and MEP processing, L2CP processing, and virtual switching. The table (ANA_CL:IPT) contains 512 entries (one per ISDX). The table is indexed with the frame's classified ISDX out of VCAP CLM. Non-service frames use entry 0. In addition to the IPT, ANA_L2 contains the ISDX table (ANA_L2:ISDX) with per-ISDX configuration related to forwarding, policing and statistics.

Each entry in the IPT provides the following settings:

- MIP index (ANA_CL:IPT:ISDX_CFG.MIP_IDX) that points to a Down-MIP half function, see Y.1731 Ethernet MIP, page 129 used by OAM Y.1731 frames.
- Layer 2 Control Protocol profile (ANA_CL:IPT:ISDX_CFG.L2CP_IDX), see Layer 2 Control Protocol Processing, page 128.
- VSI configuration (ANA_CL:IPT:VSI_CFG). The service can select to use a virtual switching instance (VSI) for forwarding. The VSI is represented by a 7-bit identifier. The VSI is used in the analyzer Layer 3 and rewriter.
- MEP configuration (ANA_CL:IPT:OAM_MEP_CFG). The service can assign one of the hardware MEPs for OAM processing. In addition, it is possible to enforce that the MEP treats all frames belonging to the service as data frames (ANA_CL:IPT:OAM_MEP_CFG.MEP_IDX_ENA).
- Ingress protection configuration (ANA_CL:IPT:IPT).

3.14.4.1 Ingress Protection

The ingress protection allows service to be protected. Each service is member of a protection group, which is configured to either use the working entity or the protection entity. When an entity fails, all services mapping to the same protection group are protected by only reconfiguring the protection group.

The analyzer classifier supports 64 protection groups (ANA_CL:PPT). Each service points to one of the protection groups through the protection group index in IPT (ANA_CL:IPT:IPT.PPT_IDX). Each service defines how the ingress protection applies (ANA_CL:IPT:IPT.IPT_CFG):

- **No protection.** The service is not protected, and the frame forwarding is not changed by the state of the protection group.
- **Upstream protection (UNI to protected NNI).** If the service's protection group uses the protection entity, then IFH field IFH.ENCAP.PROT_ACTIVE is set to 1. If the protection group uses the working entity, the field is cleared. The field IFH.ENCAP.PROT_ACTIVE is used in the rewriter by the VCAP_ES0 lookup. It enables that different encapsulations can be used whether the working or the protection entity is being used.
- **Downstream protection, working (Protected NNI to UNI).** In this mode, the service belongs to the working entity. If the associated protection group uses the protection entity, then the frame is discarded.
- **Downstream protection, protect (Protected NNI to UNI).** In this mode, the service belongs to the protection entity. If the associated protection group uses the working entity, then the frame is discarded.

A frame discarded by the protection is assigned the pipeline point configured in ANA_CL:IPT:IPT.PROT_PIPELINE_PT. The location affects the further frame processing, especially whether the frame is handled by the hardware MEPs.

3.14.5 Layer 2 Control Protocol Processing

The analyzer classifier contains a Layer 2 control protocol table (ANA_CL:L2CP_TBL) that enables EVC-based L2CP handling, for example, per EVC. The table contains 75 L2CP profiles with 64 profiles selectable from IPT (ANA_CL:IPT:ISDX_CFG.L2CP_IDX) for service frames (ISDX > 0), and 11 profiles selectable per ingress port for non-service frames (ISDX = 0). If the L2CP_IDX is 0 then the port-default profile is used.

Each profile defines the L2CP processing of BPDU and GXRP frames in terms of forwarding (tunnel, peer, discard) and COSID. The L2CP handling is defined for each of the in total 32 reserved addresses for BPDU and GXRP frames.

Each profile contains 32 entries that define the L2CP processing of BPDU and GXRP frames. The profile entries are laid out the following way:

- Entries 0-15 are used BPDU frames with destination MAC addresses in the range 0x0180C2000000 through 0x0180C200000F
- Entries 16-31 are used by GXRP frames with destination MAC addresses in the range 0x0180C2000020 through 0x0180C200002F.

Each entry in the profile defines the L2CP handling for the corresponding L2CP frames (ANA_CL:L2CP_TBL:L2CP_ENTRY_CFG). The following properties can be affected:

- Forwarding (tunnel, peer, discard) and CPU extraction queue when forwarding to the CPU.
- COSID. If enabled the frame's classified COSID out of VCAP CLM is overruled with a new value.

Note that the L2CP profile is only applied if the frame has not already been redirected by the CPU or discarded by the basic classifier or VCAP CLM. If L2CP processing of BPDU and GXRP frames is intended using the L2CP table, then CPU forwarding of BPDU and GXRP frames in the basic classifier cannot be used at the same time.

Frames that are CPU redirected or discarded by the L2CP handling are given pipeline point position ANA_CLM. An implication of this is that the L2CP table cannot be used for peering L2CP frames at a location after the hardware OAM MEPs. An example of this is L2CP frames arriving from a path at an NNI where they frames should pass transparently through the path MEP before being peered or discarded. VCAP CLM or VCAP IS2 should be used instead to peer or discard such frames.

3.14.6 Y.1731 Ethernet MIP

The analyzer classifier contains a MIP table with 128 entries (ANA_CL:MIP_TBL). Each of these entries can be used to create a Down-MIP half function. Up-MIP half functions are created by similar functionality in the rewriter.

The task of the MIP is to correctly extract relevant OAM PDUs from the frame flow for further MIP SW processing. The MIP reacts to its own MEL - it does not filter frames with other MELs. The MIP handles the OAM functions CCM, LBM, LTM, and RAPS. In addition, one generic OAM function can be configured.

The MIP to be used by a frame is pointed to by the IPT table (ANA_CL:IPT:ISDX_CFG.MIP_IDX). All 128 values can be used. The MIP is enabled by VCAP CLM action MIP_SEL. In addition to enabling the MIP, VCAP CLM must indicate under how many VLAN tags the OAM PDUs are located using VCAP CLM action OAM_Y1731_SEL. Only OAM Y.1731 frames with the correct number of VLAN tags are considered for the MIP. This check for correct number of VLAN tags is optional.

In addition, the MIP itself contains a VID filter with the following options (ANA_CL::MIP_CL_VID_CTRL.VID_SEL):

- Allow all frames that VCAP CLM has identified as MIP.
- Accept only untagged frames.
- Accept only single tagged frames with outer VID = ANA_CL::MIP_CL_VID_CTRL.VID_VAL.
- Accept both untagged frames and frames with outer VID = ANA_CL::MIP_CL_VID_CTRL.VID_VAL.

The MIP performs the following checks on VLAN-tag-valid OAM Y.1731 frames:

- Frame's MEL must match the MIP MEL (ANA_CL:MIP_TBL:MIP_CFG.MEL_VAL).

If this check passes, the frame's opcode is matched against the opcodes known to the MIP (CCM, LBM, LT, and RAPS). The MIP can recognize one more opcode with a programmable value (ANA_CL:MIP_TBL:MIP_CFG.GENERIC_OPCODE_VAL). Depending on the opcode, the following processing takes place:

- **CCM.** If CCM processing is enabled (ANA_CL:MIP_TBL:MIP_CFG.CCM_COPY_ENA), CCM frames are copied to the MIP CPU extraction queue.
- **LBM.** If LBM processing is enabled (ANA_CL:MIP_TBL:MIP_CFG.LBM_REDIR_ENA), the frame's DMAC is matched against a MIP-configured DMAC (ANA_CL:MIP_TBL:LBM_MAC_HIGH, ANA_CL:MIP_TBL:LBM_MAC_LOW). If the DMAC matches, the LBM frames are redirect to the MIP CPU extraction queue.
- **LTM.** If LTM processing is enabled (ANA_CL:MIP_TBL:MIP_CFG.LTM_REDIR_ENA), LTM frames are redirected to the MIP CPU extraction queue.
- **RAPS.** If RAPS processing is enabled (ANA_CL:MIP_TBL:MIP_CFG.RAPS_CFG), RAPS frames are either copied or redirected to the MIP CPU extraction queue, or discarded.
- **Generic Opcode.** Frames with an opcode matching the generic opcode are handled according to the configuration of ANA_CL:MIP_TBL:MIP_CFG.GENERIC_OPCODE_CFG and are either copied or redirected to the MIP CPU extraction queue, or discarded.

The MIP CPU extraction queue is configured in ANA_CL:MIP_TBL:MIP_CFG.CPU_MIP_QU. Note that if the MIP pipeline point is set to ANA_IN_MIP, a CPU copy is generated even when the pipeline point is never reached due to later processing in for instance the VOP.

By default, all CCM frames hitting a MIP entry with CCM processing enabled are redirected to the CPU. To reduce the rate of CCM frames to the CPU, CCM frames can optionally be filtered using the Hit-Me-Once (HMO) functionality available to each MIP entry. The ANA_CL::CCM_HMO_CTRL.CCM_COPY_ONCE_ENA bit can be used to do a one-time redirection of the next CCM frame processed by the MIP. When a CCM frame hits a MIP entry where the CCM_COPY_ONCE_ENA is set, the bit is automatically cleared and no further CCM frames are redirected to the CPU for this MIP entry until the bit is set again.

To ease the setting of the Hit-Me-Once bits for multiple MIP entries, each MIP entry contains a CCM internal (ANA_CL::CCM_HMO_CTRL.CCM_INTERVAL), which is a value between 0 and 3. The CCM interval allows a CPU to initiate that the CCM_COPY_ONCE_ENA bit is set for all MIP entries with a specific CCM interval:

- Set up, which CCM intervals to affect in ANA_CL::MIP_CTRL.MIP_CCM_INTERVAL_MASK. Each bit in the mask corresponds to one of the CCM intervals.
- Initiate the automatic setting of the Hit-Me-Once bits by setting ANA_CL::MIP_CTRL.MIP_CCM_HMO_SET_SHOT. This updates the Hit-Me-Once bit in all MIP entries with a CCM interval corresponding to the settings in ANA_CL::MIP_CTRL.MIP_CCM_INTERVAL_MASK.

Using the CCM intervals, a CPU can implement up to four different CCM rates, for which CCM frames are redirected to the CPU.

If one or more MIP checks fail, the MIP processing is terminated and the normal processing of the frame continues. If all checks pass, the associated CPU action is applied.

The MIP can be located in one of two pipeline point positions: ANA_OU_MIP or ANA_IN_MIP. The location affects the interaction with hardware MEPs and other analyzer functions for frames that are discarded or redirected to the CPU by the MIP. The MIP pipeline point position is configured in ANA_CL:MIP_TBL:MIP_CFG.PIPELINE_PT.

3.14.7 Analyzer Classifier Diagnostics

The analyzer classifier contains a large set of sticky bits that can inform about the frame processing and decisions made by ANA_CL. The following categories of sticky bits are available.

- Frame acceptance filtering (ANA_CL::FILTER_STICKY, ANA_CL::VLAN_FILTER_STICKY).
- VLAN and QoS classification (ANA_CL::CLASS_STICKY)
- CPU forwarding (ANA_CL::CAT_STICKY)
- MPLS processing and classification (ANA_CL::ADV_CL_MPLS_STICKY)
- VCAP CLM lookups and actions, QoS mapping tables (ANA_CL::ADV_CL_STICKY)
- MIP processing (ANA_CL::MIP_STICKY)
- IP header checks (ANA_CL::IP_HDR_CHK_STICKY)

The sticky bits are common for all ports in the analyzer classifier. All sticky bits, except for VCAP CLM sticky bits, have four corresponding sticky mask bits (ANA_CL:STICKY_MASK) that allow any sticky bit to be counted by one of the four per-port analyzer access control counters. For more information, see Analyzer Statistics, page 207.

3.15 VLAN and MSTP

The VLAN table and the MSTP table are the two main tables that control VLAN and Spanning Tree frame processing.

The following table shows the configuration parameters (located within the ANA_L3 configuration target) for each entry in the VLAN table.

Table 66 • **VLAN Table (4224 Entries)**

Field in ANA_L3:VLAN	Bits	Description
VMID	5	VMID, identifying VLAN's router leg.
VLAN_MSTP_PTR	7	Pointer to STP instance associated with VLAN.

Table 66 • **VLAN Table (4224 Entries) *(continued)***

Field in ANA_L3:VLAN	Bits	Description
VLAN_FID	13	FID to use for learning and forwarding.
VLAN_IGR_FILTER_ENA	1	Enable VLAN ingress filtering.
VLAN_SEC_FWD_ENA	1	Enables secure forwarding on a per VLAN basis. When secure forwarding is enabled, only frames with known SMAC are forwarded.
VLAN_FLOOD_DIS	1	Disables flooding of frames with unknown DMAC on a per VLAN basis.
VLAN_LRN_DIS	1	Disables learning of SMAC of frames received on this VLAN.
VLAN_RLEG_ENA	1	Enables router leg in VLAN.
VLAN_PRIVATE_ENA	1	Enables/disables this VLAN as a Private VLAN (PVLAN).
VLAN_MIRROR_ENA	1	VLAN mirror enable flag. If this field is set, frames classified to this ingress VLAN are mirrored. Note that a mirroring probe must also be configured to enable VLAN based mirroring. see Mirroring, page 218.
VLAN_PORT_MASK VLAN_PORT_MASK1	11	Specifies mask of ports belonging to VLAN.
TUPE_CTRL	16	Controls value for Table Update Engine (TUPE).

The following table shows the configuration parameters for each entry in the MSTP table.

Table 67 • **MSTP Table (66 Entries)**

Field in ANA_L3:MSTP	Bits	Description
MSTP_FWD_MASK	1 per port	Enables/disables forwarding per port
MSTP_LRN_MASK	1 per port	Enables/disables learning per port

The following table lists common parameters that also affect the VLAN and MSTP processing of frames.

Table 68 • **Common VLAN and MSTP Parameters**

Register/Field in ANA_L3_COMMON	Bits	Description
VLAN_CTRL.VLAN_ENA	1	Enables/disables VLAN lookup
PORT_FWD_CTRL	1 per port	Configures forwarding state per port
PORT_LRN_CTRL	1 per port	Configures learning state per port
VLAN_FILTER_CTRL	1 per port	Configures VLAN ingress filtering per port
VLAN_ISOLATED_CFG	1 per port	Configures isolated port mask
VLAN_COMMUNITY_CFG	1 per port	Configures community port mask

If VLAN support is enabled (in VLAN_CTRL.VLAN_ENA), the classified VID is used to lookup the VLAN information in the VLAN table. The VLAN table in turn provides an address (VLAN_MSTP_PTR) into the MSTP table. Learn, mirror, and forwarding results are calculated by combining information from the VLAN and MSTP tables.

3.15.1 Private VLAN

In a Private VLAN (PVLAN), ports are configured to be one of three different types: Promiscuous, isolated, or community.

Promiscuous Ports A promiscuous port can communicate with all ports in the PVLAN, including the isolated and community ports.

Isolated Ports An isolated port has complete Layer 2 separation from the other ports within the same PVLAN, but not from the promiscuous ports. PVLANs block all traffic to isolated ports except traffic from promiscuous ports. Traffic from isolated port is forwarded only to promiscuous ports.

Community Ports Community ports communicate among themselves and with the PVLAN's promiscuous ports. Community ports cannot communicate with isolated ports.

PVLAN can be enabled per VLAN, and port type can be configured per physical port.

3.15.2 VLAN Pseudo Code

The pseudo code for the ingress VLAN processing is shown below. For routed frames, an egress VLAN processing also takes place. This is not part of the following pseudo code.

In the pseudo code, "csr." is used to denote configuration parameters, "req." is used to denote input from previous processing steps in the analyzer, as well as information provided for the following steps in the analyzer.

```
if (req.cpu_inject != 0) {
  // cpu_inject => Bypass!
  return;
}

if (csr.vlan_ena) {
  // Get VLAN entry based on Classified VID
  igr_vlan_entry = csr.vlan_tbl[req.ivid + req.vsi_ena*4096];
} else {
  // Default VLAN entry
  igr_vlan_entry.vlan_mstp_ptr    = 0;
  igr_vlan_entry.vlan_fid         = 0;
  igr_vlan_entry.vlan_sec_fwd_ena = 0;
  igr_vlan_entry.vlan_flood_dis   = 0;
  igr_vlan_entry.vlan_lrn_dis     = 0;
  igr_vlan_entry.vlan_rleg_ena    = 0;
  igr_vlan_entry.vlan_private_ena = 0;
  igr_vlan_entry.vlan_mirror_ena  = 0;
  igr_vlan_entry.vlan_port_mask   = -1; // All-ones
  igr_vlan_entry.vmid             = 0;
}

// Retrieve MSTP state
if (csr.vlan_ena) {
  igr_mstp_entry = csr.mstp_tbl[igr_vlan_entry.vlan_mstp_ptr];
} else {
  igr_mstp_entry.mstp_fwd_mask = -1; // All-ones
  igr_mstp_entry.mstp_lrn_mask = -1; // All-ones
}

if (IsCpuPort(req.port_num) ||
    IsVD0(req.port_num) ||
    IsVD1(req.port_num)) {
  // Received from CPU or VD0/VD1 =>
  // * Do not learn
  // * Do not perform ingress filtering
  //   (these ports are not in ingress masks)
  req.l2_lrn = 0;
} else {
  // ------------------------------------------------
  // Perform ingress filtering and learning disable
  // ------------------------------------------------
```

```
// Port forwarding state
if (csr.common.port_fwd_ena[req.port_num] == 0) {
  // Ingress port not enabled for forwarding
  req.l2_fwd = 0;
  csr.port_fwd_deny_sticky =1;
}

// Port learning state
if (csr.port_lrn_ena[req.port_num] == 0) {
  req.l2_lrn = 0;
  csr.port_lrn_deny_sticky = 1;
}

if (csr.vlan_ena) {
  // Empty VLAN?
  if (igr_vlan_entry.vlan_port_mask == 0) {
    csr.vlan_lookup_invld_sticky = 1;
  }

  // VLAN ingress filtering
  if ((csr.vlan_igr_filter_ena[req.port_num] ||
       igr_vlan_entry.vlan_igr_filter_ena)
      &&
      igr_vlan_entry.vlan_port_mask[req.port_num] == 0) {
    req.l2_fwd = 0;
    req.l2_lrn = 0;
    csr.vlan_igr_filter_sticky = 1;
  } else {
    // MSTP forwarding state
    if (igr_mstp_entry.mstp_fwd_mask[req.port_num] == 0) {
      req.l2_fwd = 0;
      csr.mstp_discard_sticky = 1;
    } else {
      csr.mstp_fwd_allowed_sticky = 1;
    }

    // Learning enabled for VLAN?
    if (igr_vlan_entry.vlan_lrn_dis) {
      req.l2_lrn = 0;
      csr.vlan_lrn_deny_sticky = 1;

      // MSTP learning state
    } else if (igr_mstp_entry.mstp_lrn_mask[req.port_num] == 0) {
      req.l2_lrn = 0;
      csr.mstp_lrn_deny_sticky = 1;
    } else {
      csr.mstp_lrn_allowed_sticky = 1;
    }

    // Private VLAN handling
    if (igr_vlan_entry.vlan_private_ena) {
      if (req.vs2_avail == 0) {
        // Not received from stack port, so determine port type and
        // update req accordingly.

        // 0b00: Promiscuous port
        // 0b01: Community port
        // 0b10: Isolated port
        req.ingr_port_type = 0b00;
```

```
          if (csr.vlan_community_mask[req.port_num]) {
            req.ingr_port_type = 0b01;
          }
          if (csr.vlan_isolated_mask[req.port_num]) {
            req.ingr_port_type = 0b10;
          }
        }

        if (req.ingr_port_type == 0b10) {
          // Isolated port
          // Only allow communication with promiscuous ports
          igr_vlan_entry.vlan_port_mask &=
            (~csr.vlan_isolated_mask & ~csr.vlan_community_mask);
        } else if (req.ingr_port_type == 0b01) {
          // Community port
          // Allow communication with promiscuous ports and other community
ports,
          // but not with isolated ports
          igr_vlan_entry.vlan_port_mask &= ~csr.vlan_isolated_mask;
        }

      }
    } // vlan_igr_filter_ena
  } // csr.vlan_ena
}

// Only allow forwarding to enabled ports
igr_vlan_entry.vlan_port_mask &= csr.port_fwd_ena;

// Update req with results
req.vlan_mask = igr_vlan_entry.vlan_port_mask & igr_mstp_entry.mstp_fwd_mask;

req.ifid             = igr_vlan_entry.vlan_fid;
req.vlan_mirror      = igr_vlan_entry.vlan_mirror_ena;
req.vlan_flood_dis   = igr_vlan_entry.vlan_flood_dis;
req.vlan_sec_fwd_ena = igr_vlan_entry.vlan_sec_fwd_ena;

// Identical ingress and egress FID/VID.
// May be overruled later by egress VLAN handling.
// Note: ANA_L2 DMAC lookup always use req.efid, even when req.l3_fwd=0.
req.evid = req.ivid;
if (req.dmac[40]) {
  // If MC, DA lookup must use EVID
  req.efid = req.vsi_ena . req.evid;
} else {
  // If UC, DA lookup must use IFID
  req.efid = req.ifid;
}

// Store vmid for later use (e.g. in rleg detection)
req.irleg = igr_vlan_entry.vmid;
req.erleg = req.irleg;
```

3.15.2.1 VLAN Table Update Engine

The VLAN Table Update Engine (TUPE) can be used to quickly update a large number of port masks in the VLAN Table. For example, to support fast fail-over in scenarios with redundant forwarding paths. The following table lists the TUPE related parameters that are outside the VLAN Table.

Table 69 • **VLAN Table Update Engine (TUPE) Parameters**

Register/Field in ANA_L3:TUPE	Bits	Description
TUPE_MISC.TUPE_START	1	Start TUPE.
TUPE_MISC.TUPE_CTRL_VAL_ENA	1	Enable use of TUPE_CTRL_VAL and TUPE_CTRL_VAL_MASK.
TUPE_CTRL_BIT_ENA	1	Enable use of TUPE_CTRL_BIT_MASK.
TUPE_PORT_MASK_A_ENA	1	Enable use of TUPE_PORT_MASK_A.
TUPE_PORT_MASK_B_ENA	1	Enable use of TUPE_PORT_MASK_B
TUPE_COMB_MASK_ENA	1	Enable combined use of TUPE_CTRL_BIT_MASK and TUPE_PORT_MASK_A.
TUPE_ADDR.TUPE_START_ADDR	13	First address in VLAN table for TUPE to process.
TUPE_ADDR.TUPE_END_ADDR	13	Last address in VLAN table for TUPE to process.
TUPE_CMD_PORT_MASK_CLR TUPE_CMD_PORT_MASK_CLR1	11	TUPE command: Port mask bits to clear.
TUPE_CMD_PORT_MASK_SET TUPE_CMD_PORT_MASK_SET1	11	TUPE command: Port mask bits to set.
TUPE_CTRL_VAL	16	TUPE parameter controlling which VLAN table entries to update.
TUPE_CTRL_VAL_MASK	16	TUPE parameter controlling which VLAN table entries to update.
TUPE_CTRL_BIT_MASK	16	TUPE parameter controlling which VLAN table entries to update.
TUPE_PORT_MASK_A TUPE_PORT_MASK_A1	11	TUPE parameter controlling which VLAN table entries to update.
TUPE_PORT_MASK_B TUPE_PORT_MASK_B1	11	TUPE parameter controlling which VLAN table entries to update.

The TUPE_CTRL field in the VLAN table can be used to classify VLANs into different groups, and the VLAN_PORT_MASK for such groups of VLANs are quickly updated using TUPE. Using the parameters listed, the TUPE_CTRL field in the VLAN table can be processed as a field of individual bits, a field with one value, or a combination of the two.

For example, if a command for TUPE uses the following, the TUPE_CTRL field is treated as an 8-bit value field and eight individual bits.

- TUPE_CTRL_BIT_ENA = 1
- TUPE_CTRL_BIT_MASK = 0x00ff
- TUPE_CTRL_VAL_MASK = 0xff00

In order for TUPE to update a VLAN entry's VLAN_PORT_MASK, the value part of TUPE_CTRL must match the required value, and one or more bits of the bit part of TUPE_CTRL must be set.

If all bits in TUPE_CTRL are used as a value field, then a total of 2^{16} groups can be created, but each VLAN can only be member of one such group.

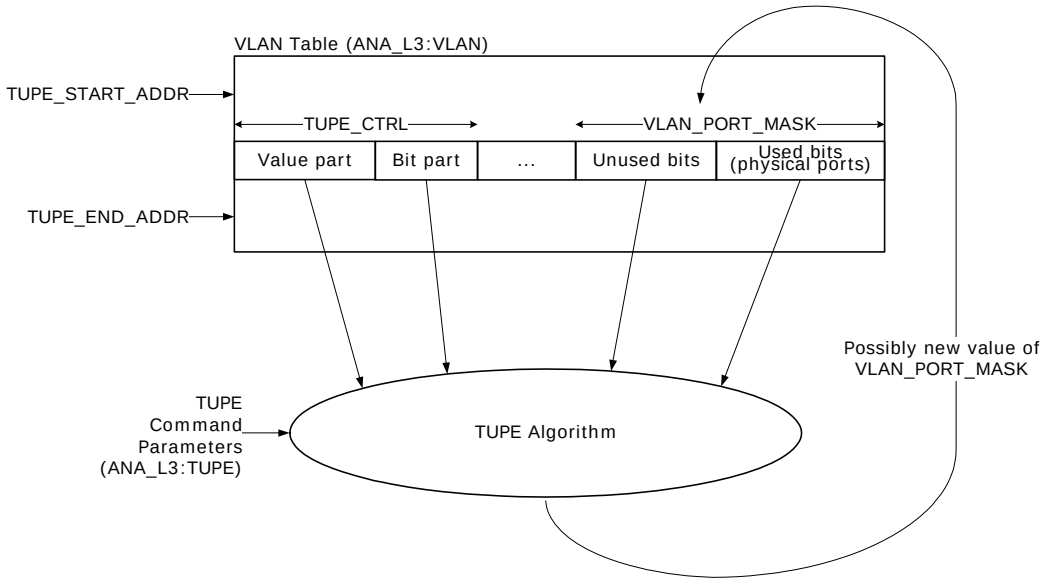
On the other end of the spectrum, if all bits in TUPE_CTRL are treated as individual bits, then only 16 groups can be created, but every VLAN can be member of any number of these groups.

In configurations that have fewer physical ports than the number of bits in the VLAN_PORT_MASK, such bits can be used as an extension to the bit part of the TUPE_CTRL field.

Apart from the VLAN's TUPE_CTRL, the value of the VLAN's VLAN_PORT_MASK is also used to decide whether to update a given VLAN table entry.

The following illustration depicts the TUPE functionality.

Figure 37 • **VLAN Table Update Engine (TUPE)**



The following pseudo code shows the full algorithm used by TUPE.

```
for (addr = csr.tupe_start_addr; addr < csr.tupe_end_addr; addr++) {
  vlan_entry = vlan_tbl[addr];
  if (
      (// If enabled, check matching value
       !csr.tupe_ctrl_val_ena
       ||
       ((vlan_entry.tupe_ctrl & csr.tupe_ctrl_val_mask) == csr.tupe_ctrl_val))
      &&
      (// If enabled, check if VLAN's tupe_ctrl has any bits overlapping
       // with csr.tupe_CTRL_BIT_MASK
       !csr.tupe_ctrl_bit_ena
       ||
       ((vlan_entry.tupe_ctrl & csr.tupe_ctrl_bit_mask) != 0))
      &&
      (// If enabled, check if VLAN's vlan_port_mask has any bits overlapping
       // with csr.tupe_PORT_MASK_A
       !csr.tupe_port_mask_a_ena
       ||
       ((vlan_entry.vlan_port_mask & csr.tupe_port_mask_a) != 0))
      &&
      (// Combined mode
       // If enabled, check if VLAN's tupe_ctrl has any bits overlapping
       // with TUPE_CTRL_BIT_MASK or VLAN's vlan_port_mask has any bits
       // overlapping with TUPE_PORT_MASK_A
       // I.e. the bit mask part of TUPE_CTRL is extended with bits from
       // vlan_port_maks.
       !csr.tupe_comb_mask_ena
       ||
       ((vlan_entry.tupe_ctrl & csr.tupe_ctrl_bit_mask) != 0)
       ||
       ((vlan_entry.vlan_port_mask & csr.tupe_port_mask_a) != 0))
      &&
      (// If enabled, check if VLAN's vlan_port_mask has any bits overlapping
       // with csr.tupe_PORT_MASK_B
```

```
        !csr.tupe_port_mask_b_ena
        ||
        ((vlan_entry.vlan_port_mask & csr.tupe_port_mask_b) != 0))
      ) {
    // vlan_port_mask must be updated for this addr
    for (p = 0; p < port_count; p++) {
      if (csr.tupe_cmd_port_mask_clr[p] == 1 &&
          csr.tupe_cmd_port_mask_set[p] == 1) {
        // clr+set => toggle
        vlan_entry.vlan_port_mask[p] = !vlan_entry.vlan_port_mask[p];
      } else if (csr.tupe_cmd_port_mask_clr[p] == 1) {
        vlan_entry.vlan_port_mask[p] = 0;
      } else if (csr.tupe_cmd_port_mask_set[p] == 1) {
        vlan_entry.vlan_port_mask[p] = 1;
      }
    }
    // Write back to VLAN table
    vlan_tbl[addr] = vlan_entry;
  }
}
```

3.16 VCAP LPM: Keys and Action

The IP addresses used for IP source/destination guard and for IP routing are stored in a VCAP in the VCAP_SUPER block. The part of the VCAP_SUPER block allocated for this purpose is VCAP LPM (Longest Path Match).

VCAP LPM encodes both the IP addresses (VCAP keys) and the action information associated with each VCAP key.

The following table shows the different key types supported by the VCAP LPM.

Table 70 • **VCAP LPM Key and Sizes**

Key Name	Key Size	Number of VCAP Words
SGL_IP4	33	1
DBL_IP4	64	2
SGL_IP6	129	4
DBL_IP6	256	8

The following table provides an overview of the VCAP LPM key. When programming an entry in VCAP LPM, the associated key fields listed must be programmed in the listed order with the first field in the table starting at bit 0 of the entry. For more information, see, Versatile Content-Aware Processor (VCAP), page 54.

Table 71 • **VCAP LPM Key Overview**

Field Name	Short Description	Size	SGL_IP4	DBL_IP4	SGL_IP6	DBL_IP6
DST_FLAG	0=SIP, 1=DIP	1	x		x	
IP4_XIP	IPv4 SIP/DIP	32	x			
IP4_SIP	IPv4 SIP	32		x		
IP4_DIP	IPv4 DIP	32		x		
IP6_XIP	IPv6 SIP/DIP	128			x	
IP6_SIP	IPv6 SIP	128				x
IP6_DIP	IPv6 DIP	128				x

3.16.1 VCAP LPM SGL_IP4 Key Details

The SGL_IP4 key is to be used for IPv4 UC routing as well as IPv4 Source/Destination Guard.

Table 72 • **VCAP LPM SGL_IP4 Key Details**

Field Name	Description	Size
DST_FLAG	0: IP4_XIP is only to be used for SIP matching. 1: IP4_XIP is only to be used for DIP matching.	1
IP4_XIP	IPv4 address	32

3.16.2 VCAP LPM DBL_IP4 Key Details

The DBL_IP4 key is to be used for IPv4 MC routing.

Table 73 • **VCAP LPM DBL_IP4 Key Details**

Field Name	Description	Size
IP4_SIP	IPv4 source address	32
IP4_DIP	IPv4 destination address	32

3.16.3 VCAP LPM SGL_IP6 Key Details

The SGL_IP6 key is to be used for IPv6 UC routing as well as IPv6 Source/Destination Guard.

Table 74 • **VCAP LPM SGL_IP6 Key Details**

Field Name	Description	Size
DST_FLAG	0: IP6_XIP is only to be used for SIP matching 1: IP6_XIP is only to be used for DIP matching	1
IP6_XIP	IPv6 address	128

3.16.4 VCAP LPM DBL_IP6 Key Details

The DBL_IP6 key is to be used for IPv6 MC routing.

Table 75 • **VCAP LPM DBL_IP6 Key Details**

Field Name	Description	Size
IP6_SIP	IPv6 source address	128
IP6_DIP	IPv6 destination address	128

3.16.5 VCAP LPM Actions

VCAP LPM supports three different actions:

- ARP_PTR
- L3MC_PTR
- ARP_ENTRY

The following table lists the actions and the key types for which they are intended to be used.

Table 76 • **VCAP LPM Action Selection**

Action Name	Size	Description and Appropriate VCAP LPM Key Types
ARP_PTR	18	Provides pointer to ARP entry in ARP Table (ANA_L3:ARP). LPM key types: SGL_IP4, SGL_IP6.
L3MC_PTR	10	Provides pointer to L3MC Table (ANA_L3:L3MC). LPM key types: DBL_IP4, DBL_IP6.
ARP_ENTRY	62	ARP entry. LPM key types: SGL_IP4, SGL_IP6.

The following table provides information about the VCAP LPM actions. When programming an action in VCAP LPM, the associated action fields listed must be programmed in the listed order, with the first field in the table starting at bit 0 of the action.

Table 77 • **VCAP LPM Actions**

Field Name	Description and Encoding	Size	ARP_PTR	L3MC_PTR	ARP_ENTRY
Action Type	0: ARP_PTR 1: L3MC_PTR 2: ARP_ENTRY 3: Reserved	2	x	x	x
ARP_PTR	Pointer to entry in ARP Table (ANA_L3:ARP)	8	x		
ARP_PTR_REMAP_ENA	If set, ARP_PTR is used to point to an entry in the ARP Pointer Remap Table (ANA_L3:ARP_PTR_REMAP).	1	x		
ECMP_CNT	Number of equal cost, multiple paths routes to DIP. See IP Multicast Routing, page 150.	4	x		
RGID	Route Group ID. Used for SIP RPF check. See SIP RPF Check, page 153.	3	x		
L3MC_PTR	Pointer to entry in L3MC Table (ANA_L3:L3MC)	8		x	
MAC_MSB	See ANA_L3:ARP:ARP_CFG_0.MAC_MSB.	16			x
MAC_LSB	See ANA_L3:ARP:ARP_CFG_1.MAC_LSB.	32			x
ARP_VMID	See ANA_L3:ARP:ARP_CFG_0.ARP_VMID.	5			x
ZERO_DMAC_CPU_QU	See ANA_L3:ARP:ARP_CFG_0.ZERO_DMAC_CPU_QU.	3			
SIP_RPF_ENA	See ANA_L3:ARP:ARP_CFG_0.SIP_RPF_ENA.	1			
SECUR_MATCH_VMID_ENA	See ANA_L3:ARP:ARP_CFG_0.SECUR_MATCH_VMID_ENA.	1			
SECUR_MATCH_MAC_ENA	See ANA_L3:ARP:ARP_CFG_0.SECUR_MATCH_MAC_ENA.	1			
ARP_ENA	See ANA_L3:ARP:ARP_CFG_0.ARP_ENA.	1			

3.17 IP Processing

This section provides information about IP routing and IP security checks. The configuration parameters for IP routing are located within the ANA_L3 configuration target.

Each frame is subject to two lookups in VCAP LPM. One lookup is used for looking up SIP, and the other lookup is used for looking up DIP or DIP+SIP (for IP multicast frames).

3.17.1 IP Source/Destination Guard

For security purposes, the device can be configured to identify specific combinations of the following information and apply security rules based on that information.

- (SMAC, SIP) or (VMID, SIP)
- (DMAC, DIP) or (VMID, DIP)

The VCAP LPM and ARP table in ANA_L3 are used to perform matching. The result of the matching is available for security rules in ANA_ACL through the following VCAP fields:

- L3_SMAC_SIP_MATCH. Set to 1 if (VMID, SIP) and/or (SMAC, SIP) match was found.
- L3_DMAC_DIP_MATCH. Set to 1 if (VMID, DIP) and/or (DMAC, DIP) match was found.

IP source/destination guard check can be enabled per port using COMMON:SIP_SECURE_ENA and COMMON:DIP_SECURE_ENA.

When enabled, the frame's DIP and the frame's SIP are each looked up in VCAP LPM. The associated VCAP action provides an index to an ARP Table entry in which the required SMAC/DMAC and/or VMID is configured.

The following pseudo code specifies the behavior of the IP Source Guard checks.

```
// Determine value of req.l3_sip_match (available in ANA_ACL as
L3_SMAC_SIP_MATCH)

if (!req.ip4_avail && !req.ip6_avail) {
  req.l3_sip_match = 1;
  return;
}

if (csr.sip_cmp_ena(req.port_num)) {
  req.l3_sip_match = 0;
} else {
  req.l3_sip_match = 1;
}

if (!LpmHit()) {
  return;
}

if (req.ip4_avail) {
  csr.secur_ip4_lpm_found_sticky = 1;
}
if (req.ip6_avail) {
  csr.secur_ip6_lpm_found_sticky = 1;
}

sip_arp_entry = csr.arp_tbl[sip_lpm_entry.arp_ptr];

if ((sip_arp_entry.secur_match_vmid_ena == 0 ||
     igr_vlan_entry.vmid == sip_arp_entry.arp_vmid)
    &&
    (sip_arp_entry.secur_match_mac_ena == 0 ||
     req.smac == sip_arp_entry.mac)) {
```

```
  req.l3_sip_match = 1;

  if (req.ip4_avail) {
    csr.secur_ip4_sip_match_sticky = 1;
  } else {
    csr.secur_ip6_sip_match_sticky = 1;
  }
}

if (req.l3_sip_match == 0) {
  csr.secur_sip_fail_sticky = 1;
}
```

The IP Destination Guard check works in the same way as the SIP check, except for the following:

- DIP check is not performed if frame has a multicast DMAC.
- DIP check is not performed if router leg has been matched.

The following pseudo code specifies the behavior of the IP Destination Guard checks.

```
// Determine value of req.l3_sip_match (available in ANA_ACL as
L3_SMAC_SIP_MATCH)

if (!req.ip4_avail && !req.ip6_avail) {
  req.l3_sip_match = 1;
  return;
}

if (csr.sip_cmp_ena(req.port_num)) {
  req.l3_sip_match = 0;
} else {
  req.l3_sip_match = 1;
}

if (!LpmHit()) {
  return;
}

if (req.ip4_avail) {
  csr.secur_ip4_lpm_found_sticky = 1;
}
if (req.ip6_avail) {
  csr.secur_ip6_lpm_found_sticky = 1;
}

sip_arp_entry = csr.arp_tbl[sip_lpm_entry.arp_ptr];

if ((sip_arp_entry.secur_match_vmid_ena == 0 ||
     igr_vlan_entry.vmid == sip_arp_entry.arp_vmid)
    &&
    (sip_arp_entry.secur_match_mac_ena == 0 ||
     req.smac == sip_arp_entry.mac)) {
  req.l3_sip_match = 1;

  if (req.ip4_avail) {
    csr.secur_ip4_sip_match_sticky = 1;
  } else {
    csr.secur_ip6_sip_match_sticky = 1;
  }
}
```

```
if (req.l3_sip_match == 0) {
  csr.secur_sip_fail_sticky = 1;
}
```

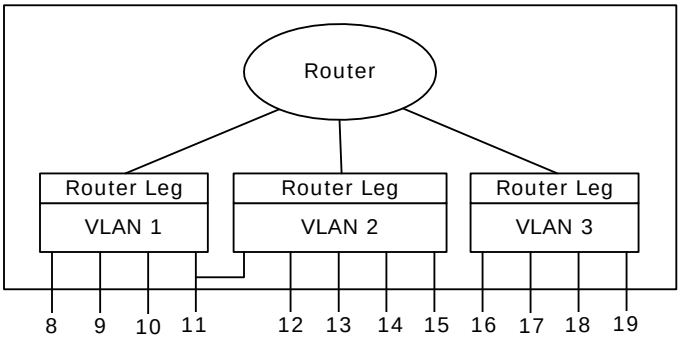
3.17.2 IP Routing

The device supports routing of IPv4 and IPv6 unicast and multicast packets through the 32 router interfaces, also known as “router legs”. Each router leg is attached to a VLAN.

The following illustration shows a configuration with three VLANs, each with an attached router leg for routing IP packets between the VLANs. Port 11 is member of both VLAN 1 and VLAN 2.

When a packet is being routed from one VLAN (the ingress VLAN) to another VLAN (the egress VLAN), the VID of the ingress VLAN is termed IVID, and the VID of the egress VLAN is termed the EVID.

Figure 38 • **Router Model**



Within the device, a router leg is identified by a 5-bit VMID value. When a packet is being routed, it is then received by the router entity using a router leg identified by an ingress VMID (or IVMID) and transmitted on an egress router leg identified by an egress VMID (or EVMID).

IP routing, also known as L3 forwarding, is only supported in VLAN-aware mode.

3.17.2.1 Frame Types for IP Routing

In order for IP packets to be routed by the device, the following conditions must be met.

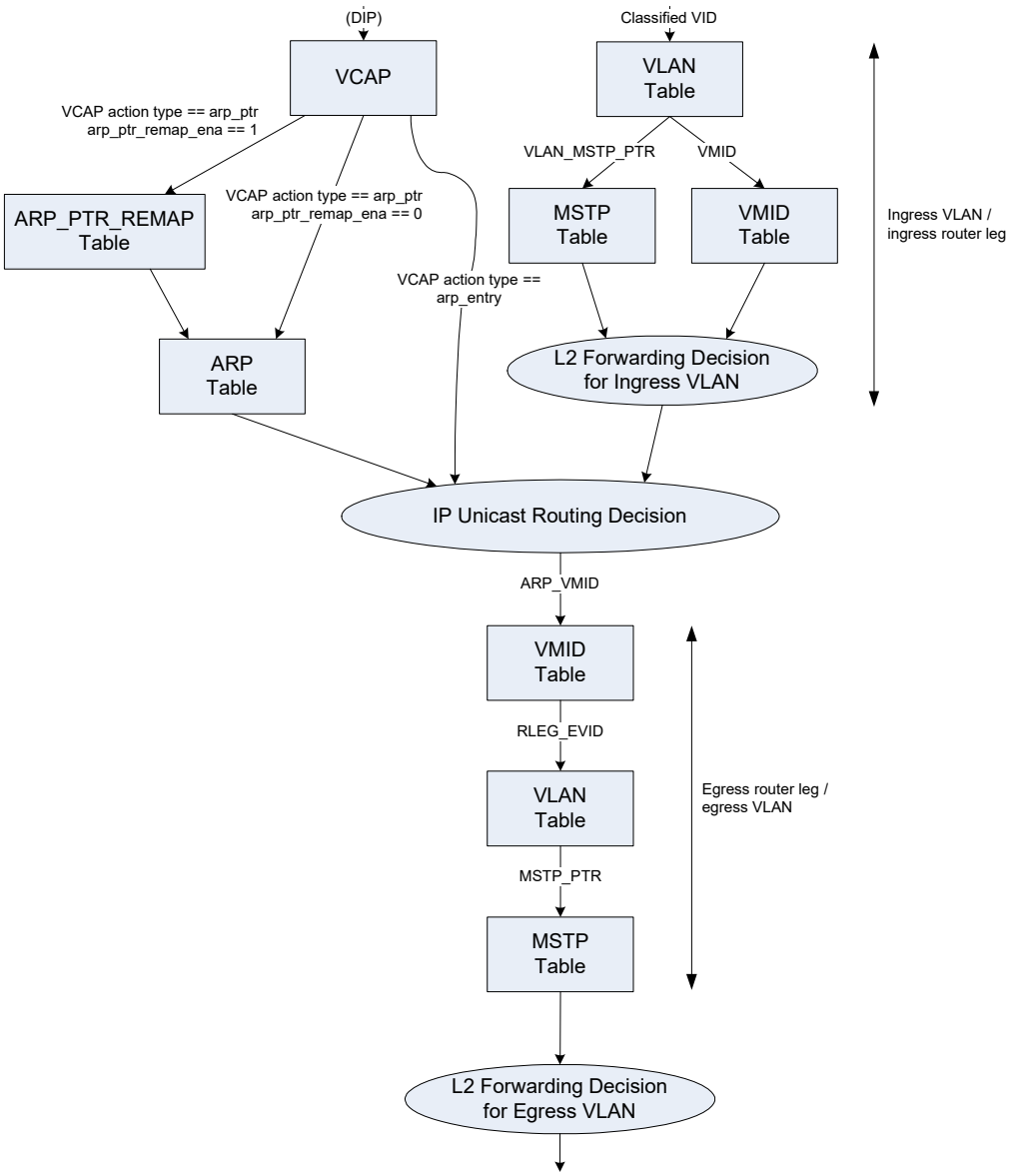
- IPv4 packets must not contain options.
- IPv6 packets must not contain hop-by-hop options.
- Optionally packets with some illegal SIP and DIP addresses can be filtered. For more information, see Illegal IPv4 Addresses, page 146 and Illegal IPv6 Addresses, page 146.
- IPv4 header checksum must be correct.

3.17.2.2 Routing Table Overview

The following illustration provides an overview of the table lookups involved in IP unicast routing within the L3 part of the analyzer.

The VCAP and ARP table lookups are performed in parallel with the VLAN/MSTP/VMID table lookups. If the frame does not match a router leg, the result of VCAP and ARP table lookups are not used for routing.

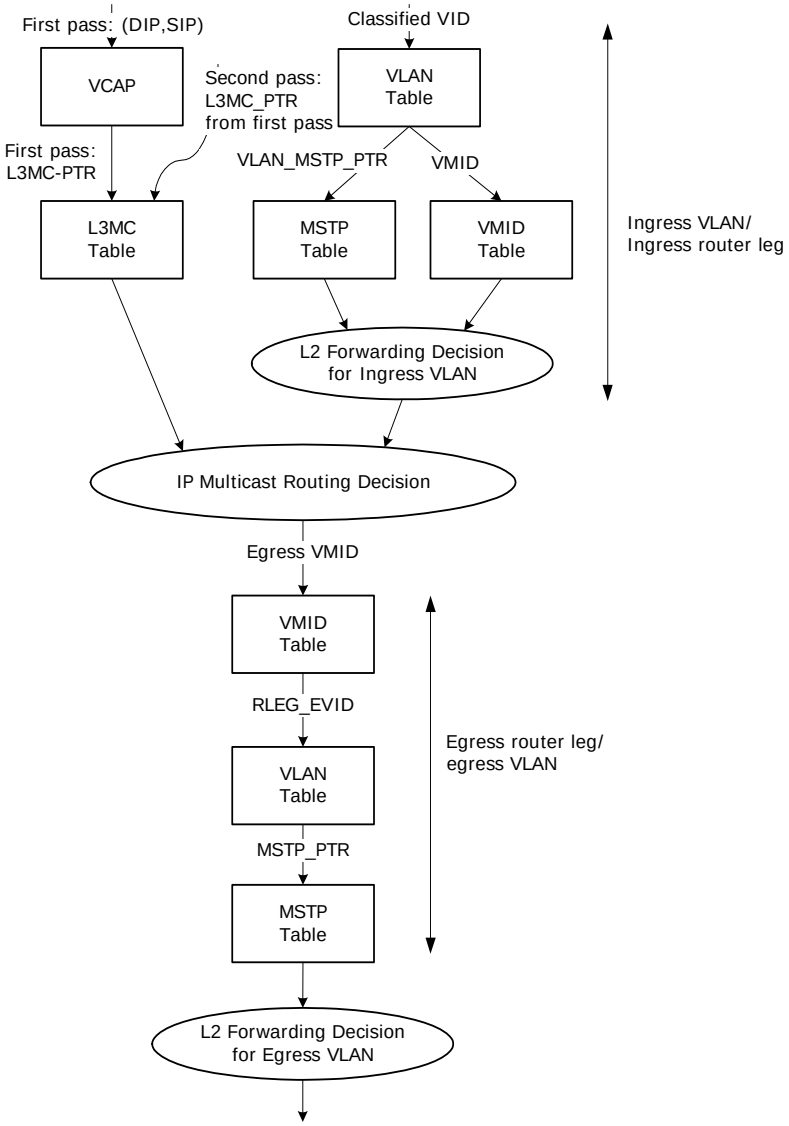
Figure 39 • **Unicast Routing Table Overview**



The following illustration provides an overview of the table lookups involved in IP multicast routing within the L3 part of the analyzer.

The VCAP and L3MC table lookups are performed in parallel with the VLAN/MSTP/VMID table lookups. If the frame does not match a router leg, the result of VCAP and L3MC table lookups are not used for routing.

Figure 40 • **Multicast Routing Table Overview**



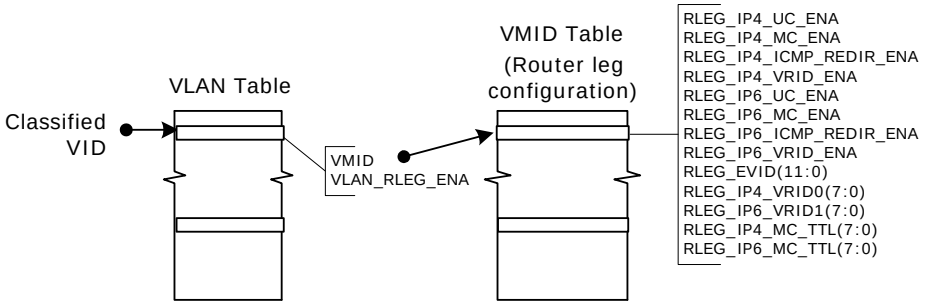
3.17.2.3 Ingress Router Leg Lookup

The VID used to determine the ingress router leg is the classified VID. The classified VID can be deducted based on a variety of parameters, such as VLAN tag or ingress port. For more information, see VLAN Classification, page 109.

A total of 32 router legs are supported. Each router leg can be used for both IPv4 and IPv6 unicast and multicast routing. A maximum of one router leg per VLAN is supported.

When processing incoming frames, the classified VID is used to index the VLAN table. The RLEG_ENA parameter determines if the VLAN is enabled for routing of packets, and if so, the VMID is used to index the VMID table. The flow is depicted in the following illustration.

Figure 41 • **Ingress Router Leg Lookup Flow**



3.17.2.3.1 Unicast Router Leg Detection

On L2, up to three different MAC addresses are used to identify the router leg:

- The normal router leg MAC address. This MAC address consists of a base address that is configured using COMMON:RLEG_CFG_0.RLEG_MAC_LSB and COMMON:RLEG_CFG_0.RLEG_MAC_MSB.

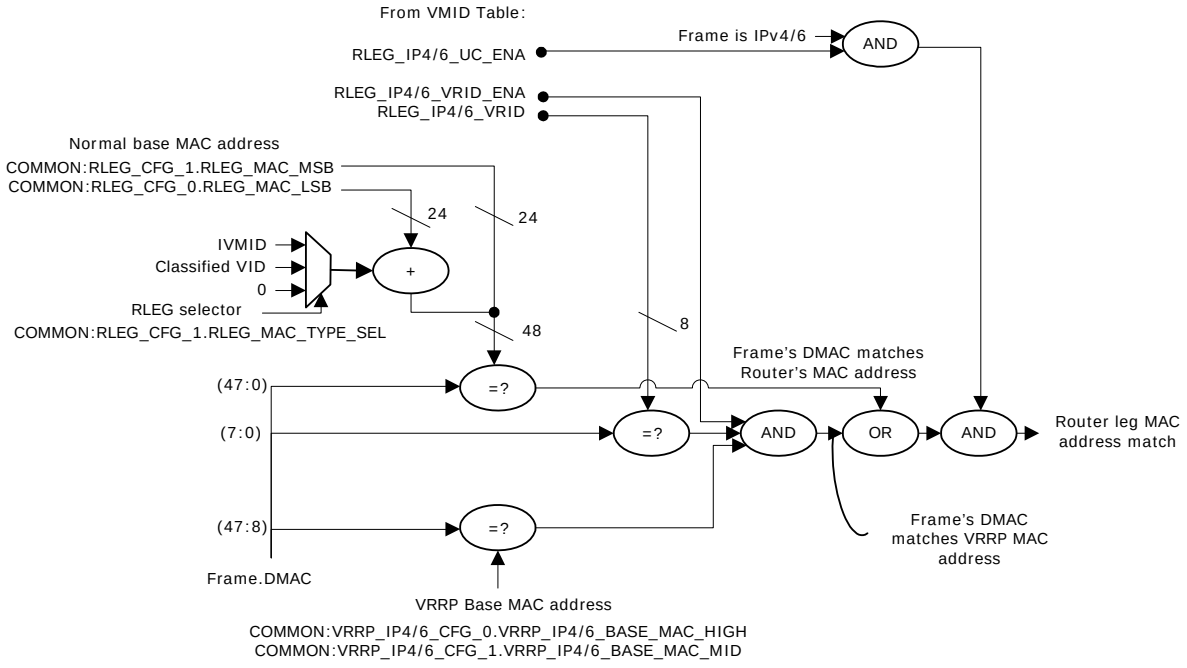
  To have different MAC addresses for each router leg, the three least significant bytes of the base MAC address can optionally be incremented by the Classified VID or the VMID. This is configured using COMMON:RLEG_CFG_1.RLEG_MAC_TYPE_SEL.
- VRRP MAC address for IPv4. This MAC address consists of a base address that is configured using COMMON:VRRP_IP4_CFG_0.VRRP_IP4_BASE_MAC_MID and COMMON:VRRP_IP4_CFG_0.VRRP_IP4_BASE_MAC_HIGH.

  For each router leg, the least signficant byte is configured per router leg using VMID:VRRP_CFG.RLEG_IP4_VRID. Up to two VRID values are supported per router leg.

  Use of VRRP for IPv4 is enabled using VMID:RLEG_CTRL.RLEG_IP4_VRID_ENA.
- VRRP MAC address for IPv6. This is configured in the same manner as VRRP MAC address for IPv4.

The matching of router leg MAC address for unicast packets is shown in the following illustration.

Figure 42 • **Ingress Router Leg MAC Address Matching for Unicast Packets**



The packet is a candidate for unicast routing if a frame's DMAC matches one of the router leg MAC addresses. Packets sent to the router itself will also match the router leg MAC address, but these will be caught through entries in the ARP table.

Despite matching a router leg MAC address, routing is not performed for any of the following reasons. If any of following criteria are met, the frame may be redirected to the CPU, depending on configuration parameters.

- It is not an IP packet (for example, an ARP reply).
- There are IP header errors.
- DIP or SIP address is illegal (optional).
- Packet contains IPv4 options/IPv6 hop-by-hop options.
- IPv4 TTL or IPv6 HL is less than two.

3.17.2.3.2 Multicast Router Leg Detection

For each router leg, IP multicast routing can be enabled using VMID:RLEG_CTRL.RLEG_IP4_MC_ENA and VMID:RLEG_CTRL.RLEG_IP6_MC_ENA.

In addition to having multicast routing enabled for the router leg configured for the VLAN, the frame's DMAC must be within a specific range in order for the packet to be candidate for IPv4/6 routing.

- IPv4 DMAC range: 01-00-5E-00-00-00 to 01-00-5E-7F-FF-FF
- IPv6 DMAC range: 33-33-00-00-00-00 to 33-33-FF-7F-FF-FF

Despite having a DMAC address within the required range, routing is not performed for any of the following reasons:

- There are IP header errors.
- DIP or SIP address is illegal (optional).
- Packet contains IPv4 options/IPv6 hop-by-hop options
- IPv4 TTL or IPv6 HL is less than two.

If any of above criteria are met, then the frame may be redirected to CPU, depending on configuration parameters.

The frame is still L2 forwarded if no router leg match is found.

3.17.2.3.3 Illegal IPv4 Addresses

Use the ROUTING_CFG.IP4_SIP_ADDR_VIOLATION_REDIR_ENA parameters to configure each of the following SIP address ranges to be considered illegal.

- 0.0.0.0 to 0.255.255.255 (IP network zero)
- 127.0.0.0 to 127.255.255.255 (IP loopback network)
- 224.0.0.0 to 255.255.255.255 (IP multicast/experimental/broadcast)

Frames with an illegal SIP can be redirected to the CPU using CPU_RLEG_IP_HDR_FAIL_REDIR_ENA.

Use the ROUTING_CFG.IP4_DIP_ADDR_VIOLATION_REDIR_ENA parameters to configure each of the following DIP address ranges to be considered illegal.

- 0.0.0.0 to 0.255.255.255 (IP network zero)
- 127.0.0.0 to 127.255.255.255 (IP loopback network)
- 240.0.0.0 to 255.255.255.254 (IP experimental)

Frames with an illegal DIP can be redirected to the CPU using CPU_RLEG_IP_HDR_FAIL_REDIR_ENA.

3.17.2.3.4 Illegal IPv6 Addresses

Use the ROUTING_CFG.IP6_SIP_ADDR_VIOLATION_REDIR_ENA parameters to configure each of the following SIP address ranges to be considered illegal.

- ::/128 (unspecified address)
- ::1/128 (loopback address)
- ff00::/8 (IPv6 multicast addresses)

Frames with an illegal SIP can be redirected to the CPU using CPU_RLEG_IP_HDR_FAIL_REDIR_ENA.

Use the ROUTING_CFG.IP6_DIP_ADDR_VIOLATION_REDIR_ENA parameters to configure each of the following DIP address ranges to be considered illegal.

- ::/128 (unspecified address)
- ::1/128 (loopback address)

Frames with an illegal DIP can be redirected to the CPU using CPU_RLEG_IP_HDR_FAIL_REDIR_ENA.

3.17.2.4 Unicast Routing

The IPv4/IPv6 unicast forwarding table in the Analyzer is configured with known routes. There are typically two types of routes: Local IP and Remote IP.

- Local IP networks directly accessible through the router. These entries return the DMAC address of the destination host and correspond to an ARP lookup.
- Remote IP networks accessible through a router. These entries return the DMAC address of the next-hop router.

If frames are deemed suitable for IP unicast routing, a DIP lookup is performed in the Longest Prefix Match (LPM) table. The lookup is a Classless Inter-Domain Routing (CIDR) lookup. That is, all network sizes are supported.

For IP addresses in local IP networks without corresponding MAC addresses, LPM entry with a corresponding ARP entry with a zero MAC address should be configured. This results in redirecting the matching frames to the CPU and thus allows the CPU to exchange ARP/NDP messages with the stations on the local IP networks to request their MAC address and afterwards use this information to update the ARP table.

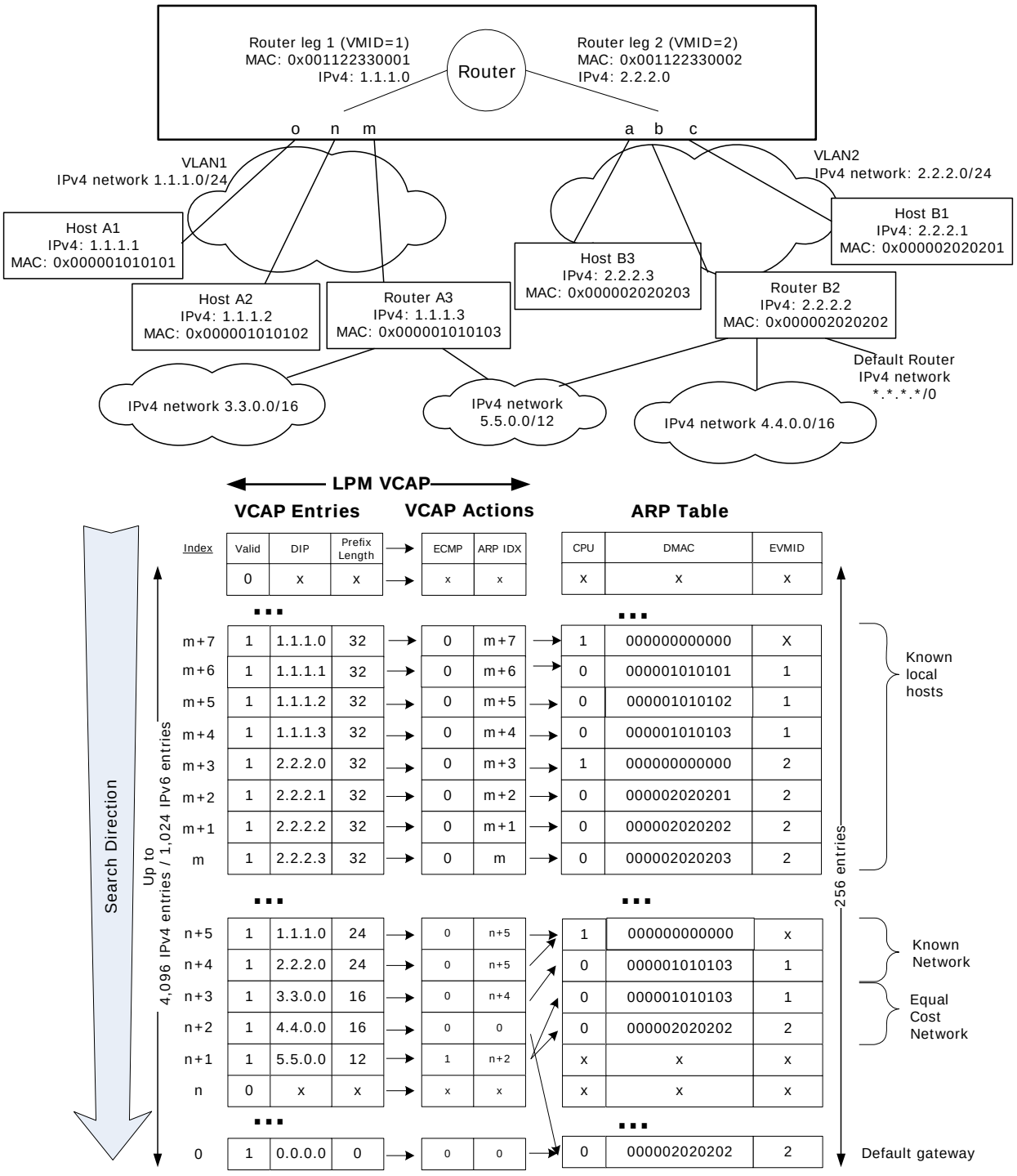
Packets with IVMID = EVMID (packets that are to be routed back to the router leg on which they were received) can optionally be redirected to the CPU such that the CPU can generate an ICMP Redirect message. This is configurable using VMID:RLEG_CTRL.RLEG_IP4_ICMP_REDIR_ENA and VMID:RLEG_CTRL.RLEG_IP6_ICMP_REDIR_ENA.

If such packets are not redirected to the CPU, they are routed.

IVMID and EVMID are configured using VLAN:VMID_CFG.VMID and ARP:ARP_CFG_0.ARP_VMID.

The following illustration shows an IPv4 unicast routing example. IPv6 unicast routing works identically to IPv4 unicast routing, with the exception that each of the IPv6 addresses occupies four times as much space in VCAP LPM.

Figure 43 • **IP Unicast Routing Example**



Ports o, n, and m are connected to Router A3, Host A1, and Host A2. These stations are included in an IP subnet associated with VLAN 1 and connected to the router by router leg 1. Router leg 1's MAC address is shown.

Ports a, b, and c are connected to Router B2, Host B1, and Host B3. These stations are included in an IP subnet associated with VLAN 2 and connected to the router by router leg 2.

The router leg MAC addresses in the example offsets the base MAC address by the respective router leg's VMID.

Traffic destined within a subnet is L2-forwarded. For example, traffic from Host A1 to Host A2 is L2 forwarded.

Traffic destined outside a subnet is sent to the router using the router leg's MAC address. The router performs the routing decision as follows:

1. **Known local host.** If the destination IP address is known to be a local host (with the full IP address installed in the VCAP LPM table), the router takes the local host DMAC address and egress router leg VMID (EVMID) from the ARP table entry pointed to by the ARP_IDX of the VCAP Action. For example, if traffic sent from Host A1 to B1 is routed using LPM entry m+2 (2.2.2.1/32) and ARP entry m + 2, the local host's DMAC address is found to be 0x000002020201 and egress router leg to be RLEG2 (that is, VMID = 2).
2. **Known longest prefix.** If the full destination IP address is not known, the router finds the longest matching IP prefix and uses it to determine the next hop DMAC address and egress router leg. For example, traffic sent from host A1 to host 4.4.4.2 is routed using LPM entry *n* + 2 (the 4.4.0.0 subnet is reached through Router B2) and using ARP entry 0, the next hop's DMAC address is found to be 0x000002020202 and egress router leg to be RLEG2 (for example, VMID = 2).
3. **Known equal cost multiple paths (ECMP).** If the full destination IP address is not known by the router and the longest matching IP prefix is an equal cost path, the next hop DMAC address and egress router leg is derived from one of up to 16 ARP entries. For example, traffic sent from host A1 to host 5.5.5.13 is routed using VCAP LPM entry *n* + 1. The corresponding VCAP action has ECMP = 1, so the ARP table entry is chosen to be either *n* + 2 or *n* + 3, based on a pseudo random number. If entry *n* + 2 is chosen, the next hop's DMAC address is found to be 0x000001010103 (Router A3) and egress router leg to be RLEG1. If entry *n* + 3 is chosen, the next hop's DMAC address is found to be 0x000002020202 (Router B2) and egress router leg to be RLEG2.
4. **Default gateway.** In order to forward packets destined to unknown subnets to a default router, a default rule can be configured for the VCAP LPM. This is illustrated in the IP Unicast Routing example as a "0.0.0.0" rule. For example, traffic sent from host A1 to host 8.9.4.2 is routed using LPM entry 0 (the default router is reached through Router B2) and ARP entry 0, the next hop's DMAC address is found to be 0x000002020202 and egress Router Leg to be RLEG2.

When routing packets, the router replaces the frame's DMAC with the ARP table's DMAC and replaces the SMAC with MAC address associated with the egress router leg. The classified VID is replaced by the egress VID (VMID:RLEG_CTRL.RLEG_EVID). The TTL/HL is decremented by 1. Note that the VMID tables (ANA_L3:VMID and REW:VMID) and the router leg MAC address must be configured consistently in the analyzer Layer 3 and rewriter.

If host A2 wants to send traffic to a host in subnet 3.3.0.0/16, the host issues an ARP request and gets a response to send traffic to router A3. Host A2 can choose to ignore this request, in which case, the router routes to router A3 in the same VLAN.

Local networks 1.1.1.0/24 and 2.2.2.0/24 are also configured. Hosts and routers located in these networks can be directly L2 accessed. This includes IP addresses 1.1.1.0 and 2.2.2.0, which are the IP addresses of the router in the respective subnets to be used for IP management traffic, for example.

3.17.2.4.1 Encoding ARP Entry in VCAP Action

Instead of using an entry in the ARP table, the ARP information can alternatively be written into the VCAP action of the LPM entry so as to not be limited by the size of the ARP table.

ARP entries written into a VCAP actions have the following limitations compared to ARP entries in the ARP table.

- ECMP cannot be supported. If there are multiple paths to a given subnet, then ARP table entries must be used.
- RGID cannot be configured. If RGIDs of SIP RPF are used, then ARP table entries must be used for such routes.
- Alternative Next Hop Configuration cannot be supported. If Alternative Next Hop Configuration is required, then ARP table entries must be used.

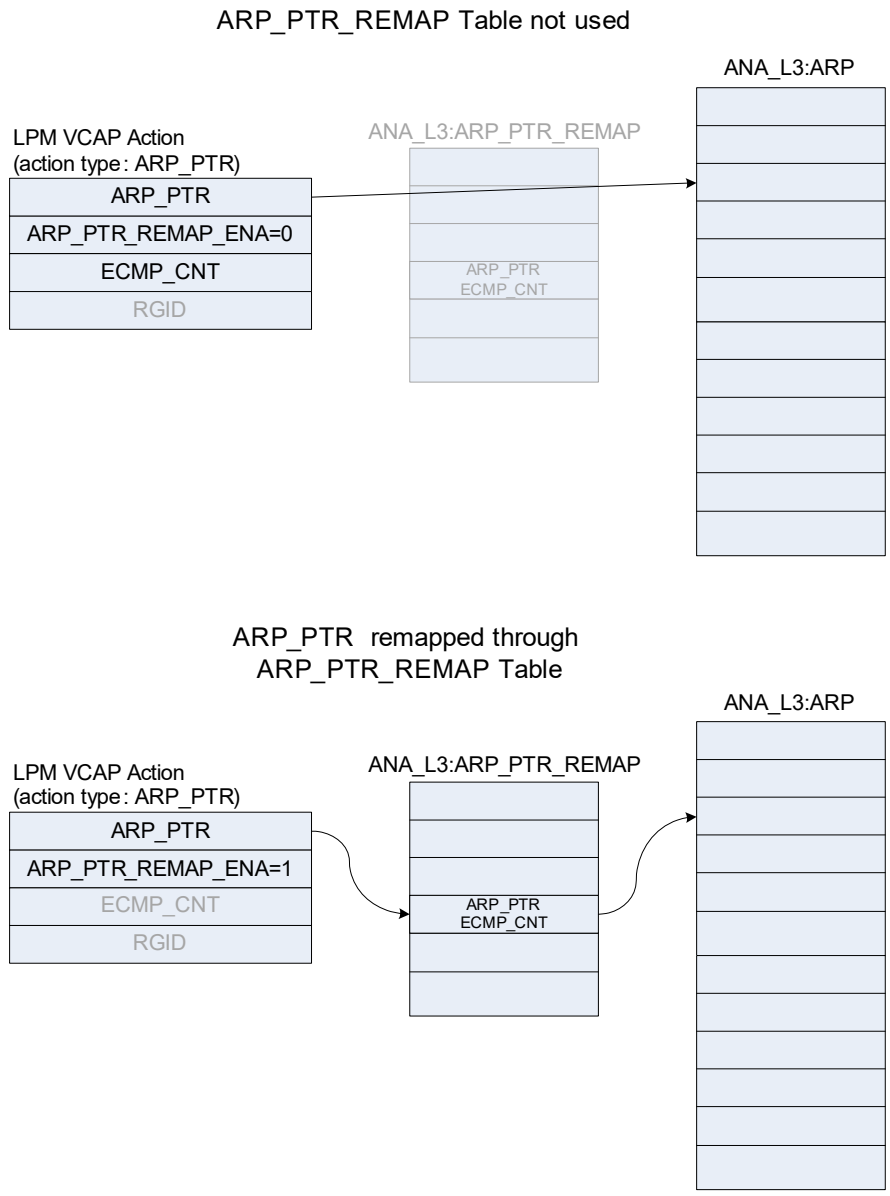
3.17.2.4.2 ARP Pointer Remap

An ARP Pointer Remap table is available to support fast fail-over when the next hop changes for a large group of LPM VCAP entries.

By setting ARP_PTR_REMAP_ENA=1 in LPM VCAP action, the ARP_PTR in the VCAP action is used to address an entry in the ARP_PTR_REMAP table and through this, new ARP_PTR and ECMP_CNT values are looked up and used for lookup in the ARP table. Thus instead of updating ARP_PTR and ECMP_CNT for many LPM VCAP entries, only the corresponding entry in ARP_PTR_REMAP needs to be updated.

The following illustration depicts both the case where ARP Pointer remapping is not used as well as the case where it is used, that is ARP_PTR_REMAP_ENA = 1.

Figure 44 • **ARP Pointer Remapping**



3.17.2.5 IP Multicast Routing

The multicast system is split into two stages, stage 1 and stage 2.

In stage 1, the frame is received on a front port and the IP multicast system in ANA_L3 is activated. This stage is used to ensure L2 forwarding to all relevant ports in the ingress VLAN as well as ingress mirroring. If the frame needs to be L3 forwarded to other VLANs, an internal port module called virtual

device 0 (VD0) also receives an L2 copy, together with information about the number of VLANs that receive a routed copy of the frame.

In stage 2, the analyzer processes each frame copy from VD0, and L3 forwards each frame copy to its egress VLAN.

In stage 1, ANA_L3 uses the VCAP LPM to determine the forwarding of the frame. Typically, two types of VCAP LPM multicast entries exist: source-specific and source-independent.

- Source specific IP multicast groups, where the same group IP address can be used by multiple sources. Such groups are denoted (S,G).
- Source independent IP multicast groups, where the group IP address uniquely identifies the IP multicast flow. Such groups are denoted (*,G).

IPv4 multicast groups occupies two entries in the VCAP LPM, and IPv6 multicast groups occupies eight entries. Source-specific multicast groups must be placed with higher precedence than source-independent multicast groups.

A matching entry in the VCAP LPM results in an index (L3MC_IDX) to an entry in the L3MC Table.

Each L3MC table entry contains a list of egress router legs to which frames are routed. Upon removing the ingress router leg from the list, the required number of routed multicast copies is calculated and this number (L3MC_COPY_CNT) is sent to VD0.

Stage 1 L2 forwards the frame. ANA_ACL can be used to limit the L2 forwarding, for example, to support IGMP/MLD snooping.

For each L3MC entry, it can optionally be configured that routing is only performed if the frame was received by a specific ingress router leg. This is termed reverse path forwarding check and is configured using L3MC:L3MC_CTRL.RPF_CHK_ENA and L3MC:L3MC_CTRL.RPF_VMID.

Note that this RPF check is unrelated to the SIP RPF check. For more information about the SIP RPF check, see SIP RPF Check, page 153.

Reverse path forwarding check does not affect the L2 forwarding decision.

In stage 2, VD0 replicates the frame according to L3MC_COPY_CNT, and L3MC_IDX is written into the frame's IFH.

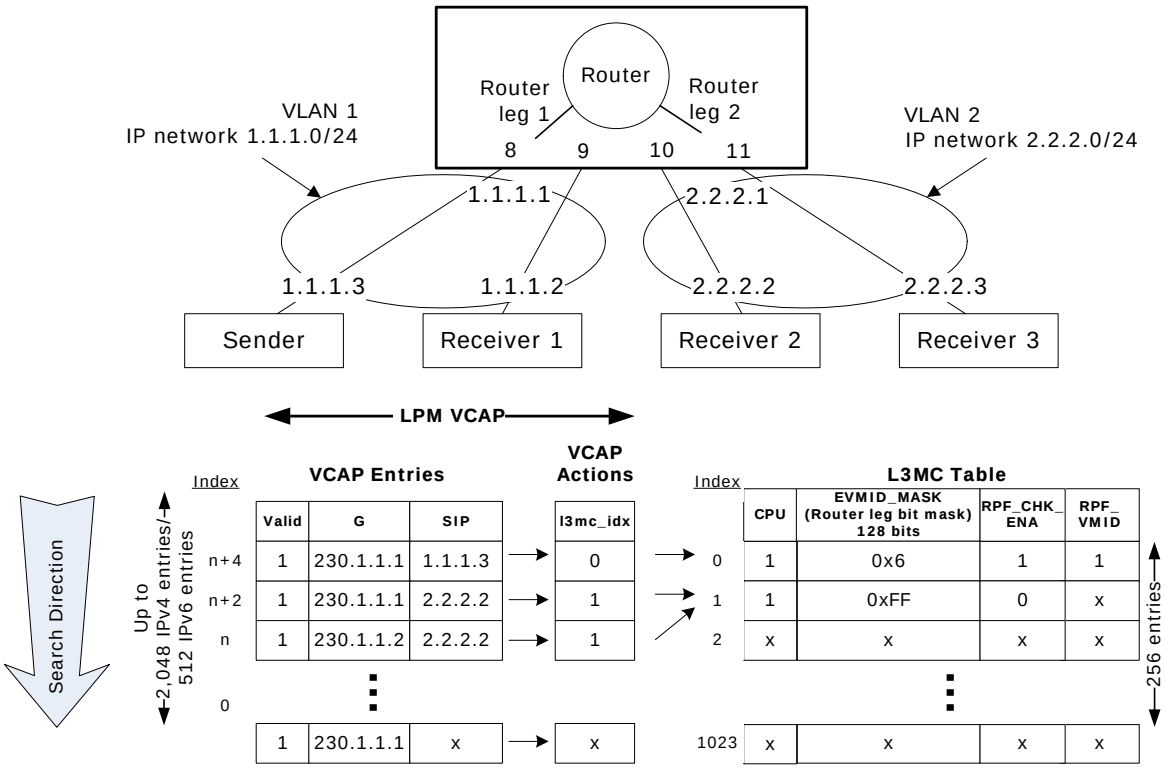
For each copy, which ANA_L3 receives from VD0 during stage 2, L3MC_IDX from IFH is used to lookup the L3MC table entry and thus find the list of egress router legs that require a routed copy. The egress router leg is used to determine the egress VID which in turn is used to perform (VID, DMAC) lookup in the MAC table for L2 forwarding within the egress VLAN.

For each egress router leg, it can be configured that frames are only L3 forwarded to the router leg if the TTL/HL of the packet is above a certain value. This is configured using VMID:VMID_MC.RLEG_IP4_MC_TTL and VMID:VMID_MC.RLEG_IP6_MC_TTL.

In stage 2, both learning and ingress filtering is disabled.

The following illustration shows an IPv4 multicast routing example. IPv6 multicast routing works identically to IPv4 multicast routing, with the exception that the IPv6 entries occupies four times as much space in VCAP LPM.

Figure 45 • **IP Multicast Routing Example**



A sender host with IPv4 address 1.1.1.3 is transmitting IPv4 multicast frames to the group address 230.1.1.1 and DMAC = 0x01005E010101. The frames must be L2 forwarded to receivers in VLAN 1 and L3 forwarded to receivers in VLAN 2. For the latter, an entry for the (1.1.1.3, 230.1.1.1) pair is added to the VCAP LPM.

3.17.2.5.1 Stage 1

- Lookup of (SIP,G) = (1.1.1.3, 230.1.1.1) in VCAP LPM returns L3MC_IDX = 0 and a corresponding entry in L3MC table. L3MC_IDX = 0 is inserted into the IFH.
- The L3MC entry has RPF check enabled (RPF_ENA = 1). Because the frame has been received through the specified router leg, L3 forwarding is allowed.
- The L3MC entry specifies that frames are forwarded to router leg 1 and 2. Because the frame has been received on router leg 1, only router leg 2 needs a routed copy. In other words, the number of L3 frame copies required is set to 1.
- If the frame's TTL is <2, then the frame is not L3 forwarded.
- Lookup of (DMAC, VID) = (0x01005E010101, 1) in the MAC table returns a destination set with port 8 and 9. Port 8 is source filtered due to normal L2 source port filtering, so the resulting destination set consists only of port 9.
- As a result of the first stage, the frame has been L2 forwarded to port 9 and a copy with L3MC_IDX = 0 and L3 copies required = 1 has been forwarded to the VD0.

3.17.2.5.2 Stage 2

- A frame is received once from VD0 with L3MC_IDX = 0 and L3MC_COPY_CNT = 1.
- Lookup of L3MC_IDX = 0 in the L3MC table returns the same entry as used in stage 1. Because this is the first routed copy, the new EVMID is 2, and through lookup in the VMID table, the corresponding EVID is 2.
- If Frame.IP.TTL < VMID.VMID_MC.RLEG_IP4_MC_TTL, the frame is not L3 forwarded.
- The frame's TTL is decremented.
- Lookup of (DMAC, VID) = (0x01005E010101, 2) in the MAC table returns a destination set with port 10 and 11. Because this is a routed copy, no source port filtering is applied.
- The result is a routed copy to ports 10 and 11 with replaced SMAC and VID from VMID = 2 and TTL decremented.

3.17.2.6 SIP RPF Check

As a security measure, the device can be configured to only L3 forward frames if the SIP belongs to a known subnet and is received by one (or more) specific router legs, for example, the router leg through which the subnet is reached. This configuration can be used to filter IP address spoofing attempts.

To support SIP RPF check, each route in VCAP LPM is assigned a Route Group ID (RGID); that is, any route in the VCAP LPM belongs to exactly one Route Group. Eight RGID values are supported.

In the VMID table the set of acceptable RGIDs for each router leg is configured using RGID_MASK.

Each router leg can be configured to operate in the following SIP RPF modes.

- Disabled. No SIP RPF check is performed.
- RGID mode. Frames are only L3 forwarded if the SIP's RGID is enabled in the router leg's RGID_MASK. If the ARP information is encoded directly in the VCAP action and SIP RPF mode is set to RGID mode, then an Rleg mode check is performed instead, because an ARP entry encoded in the VCAP Action contains to RGID.
- Rleg mode. Frames are only L3 forwarded if the VMID of the ARP table entry resulting from SIP LPM lookup is identical to the ingress VMID. For example, the router leg, which the frame was received on, is also the router leg used for forwarding frames to the SIP.

  Rleg mode cannot be used for ECMP LPM entries, because each route is only accepted by one router leg. In Rleg mode, no SIP RPF check is performed if SIP is reachable using an ECMP path.
- Combined mode. This is a combination of RGID mode and Rleg mode. If the SIP LPM lookup results in an LPM entry with ECMP_CNT = 0, an Rleg mode check is performed, otherwise a RGID Mode check is performed.

RFC3704 defines a number of different SIP RPF modes, which can be supported by the device as follows:

- Strict RPF. Use Rleg mode or Combined mode to also support ECMP.
- Loose RPF. Use RGID mode and configure RGID_MASK to all-ones. For example, any RGID value is acceptable.
- Loose RPF ignoring default routes. Use RGID mode and configure a separate RGID value for the default route. Exclude this value in router leg's RGID_MASK.
- Feasible RPF. Use RGID mode, group LPM entries using RGID and enable only the accepted groups in the router leg's RGID_MASK.

Note that SIP RPF is unrelated to the L3MC RPF check enabled in L3MC:L3MC_CTRL.RPF_CHK_ENA, which validates that a given IP multicast flow (identified by (S,G) or (*,G) was received by the expected router leg.

Frames that are not routed due to SIP RPF check can optionally be redirected to CPU by configuring COMMON:ROUTING_CFG.RLEG_IP4_SIP_RPF_REDIR_ENA and COMMON:ROUTING_CFG.RLEG_IP6_SIP_RPF_REDIR_ENA.

3.17.2.7 CPU Redirection of Filtered Frames

The following table provides an overview of the available CPU queues used when copying/redirecting frames to CPU.

Table 78 • **CPU Queue Overview**

CPU Queue	Description
COMMON:CPU_QU_CFG.CPU_RLEG_QU	Non-IP unicast frames matching an ingress router leg. For example, ARP PDUs. CPU queue for IP frames with L2 broadcast DMAC, received by router leg.
COMMON:CPU_QU_CFG.CPU_RLEG_IP_OPT_QU	IPv4 frames with options and IPv6 frames with hop-by-hop option.
COMMON:CPU_QU_CFG.CPU_RLEG_IP_HDR_FAIL_QU	IPv4 frames with IP header errors.

Table 78 • **CPU Queue Overview *(continued)***

CPU Queue	Description
COMMON:CPU_QU_CFG.CPU_IP_LEN_QU	CPU queue for IPv4/IPv6 frames failing MTU check.
COMMON:CPU_QU_CFG.CPU_MC_FAIL_QU	Failed IP multicast lookup or failed RPF check.
COMMON:CPU_QU_CFG.CPU_UC_FAIL_QU	Failed IPv4/IPv6 unicast LPM lookup, invalid ARP entry (ARP_ENA=0) or failed ICMP redirect check.
COMMON:CPU_QU_CFG.CPU_IP_TTL_FAIL_QU	Frames with TTL/HL <2.
COMMON:CPU_QU_CFG.CPU_SIP_RPF_QU	Frames failing SIP RPF check.
ARP:ARP_CFG_0.DMAC0_CPU_QU	MAC address in ARP entry is all-zeros.
L3MC:L3MC_CTRL.CPU_QU	L3MC.L3MC_CTRL.CPU_REDIR_ENA = 1.

3.17.3 Statistics

For each frame, ANA_L3 generates a number of events that can be counted in ANA_AC's statistics block. Events are generated separately for ingress and egress router legs, and in ANA_AC, event counting is further split into IPv4 and IPv6.

3.17.3.1 Ingress Router Leg Statistics

The ingress router leg statistics events generated by ANA_L3 are listed in the following table.

Table 79 • **Ingress Router Leg Events**

Event Name	Description
ivmid_ip_uc_received	IP unicast frame received by router.
ivmid_ip_mc_received	IP multicast frame received by router.
ivmid_ip_uc_routed	IP unicast frame received and L3 forwarded by router.
ivmid_ip_mc_routed	IP multicast frame received and L3 forwarded by router.
ivmid_ip_mc_rpf_discarded	IP multicast frame received by router, but discarded due to MC RPF check.
ivmid_ip_ttl_discarded	IP multicast frame received by router, but discarded due to TTL <2.
ivmid_ip_acl_discarded	IP frame received by router, but discarded by ACL rules in ANA_AC.

For counting of ingress router leg events, eight counter pairs are provided per router leg. Each pair consists of a counter for IPv4 and a counter for IPv6, for a total of 16 counters that are available per router leg.

For each of the eight counter pairs, an event mask can be configured to specify which of the Ingress router leg events listed will trigger the counter. Each counter pair can be configured to count either bytes or frames.

An example usage of the counters is shown in the following table.

Table 80 • **Ingress Router Leg Statistics Example**

			Event Name						
Reference	**Counter Name**	**Byte (b/)/Frame (f)**	**ivmid_ip_uc_received**	**ivmid_ip_mc_received**	**ivmid_ip_uc_routed**	**ivmid_ip_mc_routed**	**ivmid_ip_mc_rpf_discarded**	**ivmid_ip_ttl_discarded**	**ivmid_ip_acl_discarded**
RFC 4293	ipIfStatsInReceives	f	x	x					
RFC 4293	ipIfStatsInOctets	b	x	x					
RFC 4293	ipIfStatsInForwDatagrams	f			x	x			
RFC 4293	ipIfStatsInMcastPkts	f		x					
RFC 4293	ipIfStatsInMcastOctets	b		x					
RFC 4293	ipIfStatsInHdrErrors	f						x	
	MC RPF discards	f					x		
	ACL discards	f							x

3.17.3.2 Egress Router Leg Statistics

The egress router leg statistics events generated by ANA_L3 are listed in the following table.

Table 81 • **Egress Router Leg Events**

Event Name	Description
evmid_ip_uc_routed	IP unicast frame L3 forwarded by router.
evmid_ip_mc_routed	IP multicast frame L3 forwarded by router.
evmid_ip_mc_switched	IP multicast frame L2 forwarded by switch.
evmid_ip_mc_ttl_discarded	IP multicast frame discarded by router due to TTL value configured for egress router leg. See VMID:VMID_MC:RLEG_IP4_MC_TTL and VMID:VMID_MC:RLEG_IP6_MC_TTL.
evmid_ip_acl_discarded	IP frame discarded by ACL rules in ANA_AC.

For counting of egress router leg events, eight counter pairs are provided per router leg. Each pair consists of a counter for IPv4 and a counter for IPv6, for a total of 16 counters available per router leg.

For each of the eight counter pairs, an event mask can be configured to specify which of the events listed in the following table will trigger the counter. Each counter pair can be configured to count either bytes or frames.

An example usage of the counters is shown in the following table.

Table 82 • **Egress Router Leg Statistics Example**

			Event Name				
Reference	**Counter Name**	**Byte (b/)/Frame (f)**	**evmid_ip_uc_routed**	**evmid_ip_mc_routed**	**evmid_ip_mc_switched**	**evmid_ip_ttl_discarded**	**evmid_ip_acl_discarded**
RFC 4293	ipIfStatsOutForwDatagrams	f	x	x			
RFC 4293	ipIfStatsOutOctets	b	x	x			
RFC 4293	ipIfStatsOutMcastPkts	f		x			
RFC 4293	ipIfStatsOutMcastOctets	b		x			
Microsemi	IP MC TTL discards	f				x	
Microsemi	ACL discards	f					x

3.17.3.3 LPM Statistics

For debugging purposes and usage monitoring, the VCAP includes a sticky bit per entry.

3.17.4 IGMP/MLD Snooping Switch

RFC4541 describes recommendations on how an IGMP/MLD snooping switch forwards IP Multicast packets.

To implement the required data forwarding rules for an IGMP/MLD snooping switch, the VCAP IS2 lookup in ANA_ACL must be used.

For each IP multicast flow, the VCAP IS2 lookup can be used to limit the forwarding within each VLAN using the following key types:

- IGMP snooping switch: IP4_VID
- MLD snooping switch: IP6_VID

Both key types include SIP, DIP, and VID such that snooping switches can be supported for IGMPv3 and MLDv2.

In the action of the VCAP rules, the following fields can be used to limit the forwarding.

- PORT_MASK. The ports to which the frame is forwarded within the VLAN.
- MASK_MODE. Should normally be set to 1 (AND_VLANMASK).

When IP multicast routing is combined with snooping, a lookup in VCAP IS2 is made for each copy of the IP multicast packet. The VID in each such lookup is the VID of the egress VLAN (of that copy). In order for the VCAP rules to use the EVID in the IP4_VID/IP6_VID key, the second VCAP lookup must be used, and ANA_ACL::VCAP_S2_CFG.SEC_ROUTE_HANDLING_ENA must be set to 1.

3.18 VCAP IS2 Keys and Actions

VCAP IS2 is part of ANA_ACL and enables access control lists using VCAP functionality. This section provides detailed information about all available VCAP IS2 keys and actions.

For information about how to select which key to use and how the VCAP IS2 action is applied, see VCAP IS2, page 167.

3.18.1 VCAP IS2 Keys

VCAP IS2 supports a number of different keys that can be used for different purposes and frame types. The keys are of type X4, X8, or X16 depending on the number of words each key uses. The keys are grouped after size as shown in the following table.

Table 83 • **VCAP IS2 Keys and Sizes**

Key Name	Key Size	Number of Words	Key Type
IP4_VID	88 bits	4 words	X4 type
MAC_ETYPE	271 bits	8 words	X8 type
ARP	210 bits		
IP4_TCP_UDP	283 bits		
IP4_OTHER	274 bits		
CUSTOM_2	284 bits		
IP6_VID	284 bits		
IP_7TUPLE	567 bits	16 words	X16 type
CUSTOM_1	546 bits		

The following table lists details for all VCAP IS2 keys and fields. When programming an entry in VCAP IS2, the associated key fields must be programmed in the listed order with the first field in the table starting at bit 0 in the entry. As an example, for a MAC_ETYPE entry, the first field X8_TYPE must be programmed at bits 3:0 of the entry, the second field FIRST must be programmed at bit 4, and so on.

Table 84 • **VCAP IS2 Key Overview**

Field Name	Description	Size	IP4_VID	MAC_ETYPE	ARP	IP4_TCP_UDP	IP4_OTHER	CUSTOM_2	IP6_VID	IP_7TUPLE	CUSTOM_1
X8_TYPE	X8 type. 0: MAC_ETYPE 3: ARP 4: IP4_TCPUDP 5: IP4_OTHER 8: CUSTOM_2 9: IP6_VID	4		x	x	x	x	x	x		
X16_TYPE	X16 type. 1: IP7_TUPLE 2: CUSTOM_1	2								x	x
FIRST	Selects between entries relevant for first and second lookup. Set for first lookup, cleared for second lookup.	1	x	x	x	x	x	x	x	x	x
PAG	Classified Policy Association Group (PAG). PAG can be used to tie VCAP CLM lookups with VCAP IS2 lookups. The default PAG value is configurable per port in ANA_CL:PORT:PORT_ID_CFG.PAG_VAL.	8	x	x	x	x	x	x	x	x	x

Table 84 • **VCAP IS2 Key Overview *(continued)***

Field Name	Description	Size	IP4_VID	MAC_ETYPE	ARP	IP4_TCP_UDP	IP4_OTHER	CUSTOM_2	IP6_VID	IP_7TUPLE	CUSTOM_1
IGR_PORT_MASK_SEL	Mode selector for IGR_PORT_MASK. 0: Default setting. 1: Set for frames received from a loopback device, i.e. LBK_DEV*. 2: Set for masqueraded frames if ANA_ACL::VCAP_S2_MISC_CTRL.MASQ_IGR_MASK_ENA == 1. A masqueraded frame is identified by the following criteria: (IFH.FWD.DST_MODE == INJECT && IFH.SRC_PORT != physical src port) || (MISC.PIPELINE_ACT == INJ_MASQ) 3: Set for the following frame types: 3a: CPU injected frames and ANA_ACL::VCAP_S2_MISC_CTRL.CPU_IGR_MASK_ENA == 1. 3b: Frames received from VD0 or VD1 and ANA_ACL::VCAP_S2_MISC_CTRL.VD_IGR_MASK_ENA == 1. 3c: Frame received on a loopback device and ANA_ACL::VCAP_S2_MISC_CTRL.LBK_IGR_MASK_SEL3_ENA == 1. If a frame fulfills multiple of above criteria, then higher value of IGR_PORT_MASK_SEL takes precedence.	2		x	x	x	x	x		x	x
IGR_PORT_MASK	Ingress port mask. IGR_PORT_MASK_SEL == 0: Each bit in the mask correspond to physical ingress port. IGR_PORT_MASK_SEL == 1: Each bit in the mask correspond to the physical port on which the frame was looped. IGR_PORT_MASK_SEL == 2: Each bit in the mask correspond to the masqueraded port. If IGR_PORT_MASK_SEL == 3: Bit 0: Physical src port == CPU0 Bit 1: Physical src port == CPU1 Bit 2: Physical src port == VD0 Bit 3: Physical src port == VD1 Bits 4:9: Src port (possibly masqueraded or looped) Bits 44:48: IFH.MISC.PIPELINE_PT Bits 49:51: IFH.MISC.PIPELINE_ACT Bits 52: Reserved	53		x	x	x	x	x		x	x
L2_MC	Set if frame's destination MAC address is a multicast address (bit 40 = 1).	1		x	x	x	x	x		x	x

Table 84 • **VCAP IS2 Key Overview *(continued)***

Field Name	Description	Size	IP4_VID	MAC_ETYPE	ARP	IP4_TCP_UDP	IP4_OTHER	CUSTOM_2	IP6_VID	IP_7TUPLE	CUSTOM_1
L2_BC	Set if frame's destination MAC address is the broadcast address (FF-FF-FF-FF-FF-FF).	1		x	x	x	x	x		x	x
SERVICE_FRM	Set if classified ISDX > 0.	1		x	x	x	x	x		x	x
L2_FWD	Set if the frame is allowed to be forwarded to front ports. L2_FWD can for instance be cleared due to MSTP filtering or frame acceptance.	1		x	x	x	x	x		x	x
VLAN_TAGGED	Set if frame was received with a VLAN tag.	1		x	x	x	x	x		x	x
VID	For ISDX = 0 (non-service frames): Classified VID. If ANA_ACL::VCAP_S2_CFG.SEC_ROUTE_HANDLING_ENA is set, VID is IRLEG VID for first lookup and ERLEG VID for second lookup for routable frames (L3_RT=1). For ISDX > 0 (service frames): By default, classified ISDX. When ANA_ACL::VCAP_S2_MISC_CTRL.PAG_FORCE_VID_ENA is set, bit 7 in the PAG controls whether VID or ISDX is used: If PAG[7] = 0: Use classified ISDX. If PAG[7] = 1: Use classified VID.	12	x	x	x	x	x	x	x	x	x
DEI	Classified DEI.	1		x	x	x	x	x		x	x
PCP	Classified PCP.	3		x	x	x	x	x		x	x
L3_SMAC_SIP_MATCH	Match found in SIP security lookup in ANA_L3. See IP Source/Destination Guard, page 140.	1	x	x		x	x	x	x	x	x
L3_DMAC_DIP_MATCH	Match found in DIP security lookup in ANA_L3. See IP Source/Destination Guard, page 140.	1	x	x		x	x	x	x	x	x
L3_RT	Set if frame has hit a router leg.	1	x	x		x	x	x	x	x	x
L2_DMAC	Destination MAC address.	48		x						x	
L2_SMAC	Source MAC address.	48		x	x					x	
ETYPE_LEN	Frame type flag indicating that the frame is EtherType encoded. Set if frame has EtherType >= 0x600.	1		x							

Table 84 • **VCAP IS2 Key Overview *(continued)***

Field Name	Description	Size	IP4_VID	MAC_ETYPE	ARP	IP4_TCP_UDP	IP4_OTHER	CUSTOM_2	IP6_VID	IP_7TUPLE	CUSTOM_1
ETYPE	Frame's EtherType Overloading: OAM Y.1731 frames: ETYPE[6:0] contains OAM MEL flags, see below. OAM_Y1731 is set to 1 to indicate the overloading of ETYPE. OAM MEL flags: Encoding of MD level/MEG level (MEL). One bit for each level where lowest level encoded as zero. The following keys can be generated: MEL=0: 0b0000000 MEL=1: 0b0000001 MEL=2: 0b0000011 MEL=3: 0b0000111 MEL=4: 0b0001111 MEL=5: 0b0011111 MEL=6: 0b0111111 MEL=7: 0b1111111 Together with the mask, the following kinds of rules may be created: - Exact match. Fx. MEL = 2: 0b0000011 - Below. Fx. MEL< = 4: 0b000XXXX - Above. Fx. MEL> = 5: 0bXX11111 - Between. Fx. 3< = MEL< = 5: 0b00XX111, where 'X' means don't care.	16		x						
IP4	Frame type flag indicating the frame is an IPv4 frame. Set if frame is IPv4 frame (EtherType = 0x800 and IP version = 4)	1				x	x			x	
L2_PAYLOAD_ETYPE	Byte 0-7 of L2 payload after Type/Len field. Overloading: OAM Y.1731 frames (OAM_Y1731 = 1): Bytes 0-3: OAM PDU bytes 0-3 Bytes 4-5: OAM_MEPID Bytes 6-7: Unused.	64		x							
L3_FRAGMENT	Set if IPv4 frame is fragmented (more fragments flag = 1 or fragments offset > 0).	1				x	x				
L3_FRAG_OFS_GT0	Set if IPv4 frame is fragmented but not the first fragment (fragments offset > 0)	1				x	x				
ARP_ADDR_SPACE_OK	Set if hardware address is Ethernet.	1			x						
ARP_PROTO_SPACE_OK	Set if protocol address space is 0x0800.	1			x						
ARP_LEN_OK	Set if Hardware address length = 6 (Ethernet) and IP address length = 4 (IP).	1			x						
ARP_TARGET_MATCH	Target Hardware Address = SMAC (RARP).	1			x						
ARP_SENDER_MATCH	Sender Hardware Address = SMAC (ARP).	1			x						

Table 84 • **VCAP IS2 Key Overview *(continued)***

Field Name	Description	Size	IP4_VID	MAC_ETYPE	ARP	IP4_TCP_UDP	IP4_OTHER	CUSTOM_2	IP6_VID	IP_7TUPLE	CUSTOM_1
ARP_OPCODE_UNKNOWN	Set if ARP opcode is none of the below mentioned.	1			x						
ARP_OPCODE	ARP opcode. Only valid if ARP_OPCODE_UNKNOWN is cleared. 0: ARP request. 1: ARP reply. 2: RARP request. 3: RARP reply.	2			x						
L3_OPTIONS	Set if IPv4 frame contains options (IP len > 5). IP options are not parsed.	1				x	x				
L3_TTL_GT0	Set if IPv4 TTL / IPv6 hop limit is greater than 0.	1				x	x			x	
L3_TOS	Frame's IPv4/6 DSCP and ECN fields.	8				x	x			x	
L3_IP4_DIP	IPv4 frames: Destination IPv4 address. IPv6 frames: Source IPv6 address, bits 63:32.	32	x		x	x	x				
L3_IP4_SIP	IPv4 frames: Source IPv4 address. IPv6 frames: Source IPv6 address, bits 31:0.	32	x		x	x	x				
L3_IP6_DIP	IPv6 frames: Destination IPv6 address. IPv4 frames: Bits 31:0: Destination IPv4 address.	128							x	x	
L3_IP6_SIP	IPv6 frames: Destination IPv6 address. IPv4 frames: Bits 31:0: Destination IPv4 address. In addition, for IP_7TUPLE and IPv4 frames: Bit 127: L3_FRAGMENT. Bit 126: L3_FRAG_OFS_GT0. Bit 125: L3_OPTIONS.	128							x	x	
DIP_EQ_SIP	Set if destination IP address is equal to source IP address.	1			x	x	x			x	
L3_IP_PROTO	IPv4 frames (IP4 = 1): IP protocol. IPv6 frames (IP4 = 0): Next header.	8					x				
TCP_UDP	Frame type flag indicating the frame is a TCP or UDP frame. Set if frame is IPv4/IPv6 TCP or UDP frame (IP protocol/next header equals 6 or 17).	1								x	
TCP	Frame type flag indicating the frame is a TCP frame. Set if frame is IPv4 TCP frame (IP protocol = 6) or IPv6 TCP frames (Next header = 6)	1				x				x	
L4_DPORT	TCP/UDP destination port. Overloading for IP_7TUPLE: Non-TCP/UDP IP frames: L4_DPORT = L3_IP_PROTO.	16				x				x	
L4_SPORT	TCP/UDP source port.	16				x				x	

Table 84 • **VCAP IS2 Key Overview *(continued)***

Field Name	Description	Size	IP4_VID	MAC_ETYPE	ARP	IP4_TCP_UDP	IP4_OTHER	CUSTOM_2	IP6_VID	IP_7TUPLE	CUSTOM_1
L4_RNG	Range mask. Range types: SPORT, DPORT, SPORT or DPORT, VID, DSCP, custom. Input into range checkers is taken from classified results (VID, DSCP) and frame (SPORT, DPORT, custom).	8				x				x	
SPORT_EQ_DPORT	Set if TCP/UDP source port equals TCP/UDP destination port.	1				x				x	
SEQUENCE_EQ0	Set if TCP sequence number is 0.	1				x				x	
L4_FIN	TCP flag FIN.	1				x				x	
L4_SYN	TCP flag SYN.	1				x				x	
L4_RST	TCP flag RST.	1				x				x	
L4_PSH	TCP flag PSH.	1				x				x	
L4_ACK	TCP flag ACK.	1				x				x	
L4_URG	TCP flag URG.	1				x				x	
L3_PAYLOAD	Payload bytes after IP header. IPv4: IPv4 options are not parsed so payload is always taken 20 bytes after the start of the IPv4 header.	96					x				
L4_PAYLOAD	Payload bytes after TCP/UDP header. Overloading for IP_7TUPLE: Non TCP/UDP frames (TCP_UDP = 0): Payload bytes 0 – 7 after IP header. IPv4 options are not parsed so payload is always taken 20 bytes after the start of the IPv4 header for non TCP/UDP IPv4 frames.	64				x				x	
OAM_CCM_CNTS_EQ0	Flag indicating if dual-ended loss measurement counters in CCM frames are used. This flag is set if the TxFCf, RxFCb, and TxFCb counters are set to all zeros.	1		x							
OAM_Y1731	Set if frame's EtherType = 0x8902. If set, ETYPE is overloaded with OAM MEL flags, See ETYPE.	1		x							
CUSTOM1	57 bytes payload starting from current frame pointer position, as controlled by VCAP CLM actions. Note that if frame_type==ETH, then payload is retrieved from a position following the Ethernet layer, for example, after DMAC, SMAC, 0 – 3 VLAN tags, and EtherType.	456									x
CUSTOM2	24 bytes payload from frame. Location in frame is controlled by VCAP CLM actions. Note that if frame_type==ETH, payload is retrieved from a position following the Ethernet layer, for example, after DMAC, SMAC, 0 – 3 VLAN tags and EtherType.	192						x			

3.18.2 VCAP IS2 Actions

VCAP IS2 supports only one action, size 196 bits, which applies to all VCAP IS2 keys. The action is listed in the following table. When programming an action in VCAP IS2, the associated action fields listed must be programmed in the listed order with the first field in the table starting at bit 0 in the action.

Table 85 • **VCAP IS2 Actions**

Action Name	Description	Size
IS_INNER_ACL	Set if VCAP IS2 action belongs to ANA_IN_SW pipeline point.	1
PIPELINE_FORCE_ENA	If set, use PIPELINE_PT unconditionally and set PIPELINE_ACT = NONE if PIPELINE_PT == NONE. Overrules previous settings of pipeline point.	1
PIPELINE_PT	Pipeline point used if PIPELINE_FORCE_ENA is set: 0: NONE 1: ANA_VRAP 2: ANA_PORT_VOE 3: ANA_CL 4: ANA_CLM 5: ANA_IPT_PROT 6: ANA_OU_MIP 7: ANA_OU_SW 8: ANA_OU_PROT 9: ANA_OU_VOE 10: ANA_MID_PROT 11: ANA_IN_VOE 12: ANA_IN_PROT 13: ANA_IN_SW 14: ANA_IN_MIP 15: ANA_VLAN 16: ANA_DONE	5
HIT_ME_ONCE	If set, the next frame hitting the rule is copied to CPU using CPU_QU_NUM. CPU_COPY_ENA overrules HIT_ME_ONCE implying that if CPU_COPY_ENA is also set, all frames hitting the rule are copied to the CPU.	1
INTR_ENA	If set, an interrupt is triggered when this rule is hit	1
CPU_COPY_ENA	If set, frame is copied to CPU using CPU_QU_NUM.	1
CPU_QU_NUM	CPU queue number used when copying to the CPU	3
CPU_DIS	If set, disable extraction to CPU queues from previous forwarding decisions. Action CPU_COPY_ENA is applied after CPU_DIS.	1
LRN_DIS	If set, autolearning is disallowed.	1
RT_DIS	If set, routing is disallowed.	1
POLICE_ENA	If set, frame is policed by policer pointed to by POLICE_IDX. Overloading: If cleared, POLICE_IDX can be used to select a new source IP address from table ANA_ACL::SWAP_SIP when replacing IP addresses. See ACL_RT_MODE.	1
POLICE_IDX	Selects policer index used when policing frames. Overloading: If POLICE_ENA=0, selects source IP address when replacing IP addresses.	5
IGNORE_PIPELINE_CTRL	Ignore ingress pipeline control. This enforces the use of the VCAP IS2 action even when the pipeline control has terminated the frame before VCAP IS2.	1

Table 85 • **VCAP IS2 Actions *(continued)***

Action Name	Description	Size
LOG_MSG_INT	Controls logMessageInterval when REW_CMD[5] = 1: For multicast Delay_Resp frames, SWAP_MAC_ENA and LOG_MSG_INT encode a new logMessageInterval: SWAP_MAC_ENA=0, LOG_MSG_INT=0: logMessageInterval = 0 SWAP_MAC_ENA=0, LOG_MSG_INT=1: logMessageInterval = 1 ... SWAP_MAC_ENA=0, LOG_MSG_INT=7: logMessageInterval = 7 SWAP_MAC_ENA=1, LOG_MSG_INT=0: logMessageInterval = –8 SWAP_MAC_ENA=1, LOG_MSG_INT=1: logMessageInterval = –7 ... SWAP_MAC_ENA=1, LOG_MSG_INT=7: logMessageInterval = –1	3
MASK_MODE	Controls how PORT_MASK is applied. 0: OR_DSTMASK: Or PORT_MASK with destination mask. 1: AND_VLANMASK: And PORT_MASK with VLAN mask. 2: REPLACE_PGID: Replace PGID port mask from MAC table lookup by PORT_MASK. 3: REPLACE_ALL: Use PORT_MASK as final destination set replacing all other port masks. Note that with REPLACE_ALL, the port mask becomes sticky and cannot be modified by later processing steps. 4: REDIR_PGID: Redirect using PORT_MASK[7:0] as PGID table index. See PORT_MASK for extra configuration options. options. Note that the port mask becomes sticky and cannot be modified by subsequent processing steps. 5: OR_PGID_MASK: Or PORT_MASK with PGID port mask from MAC table lookup. 6: Reserved. 7: Reserved. The CPU port is untouched by MASK_MODE.	3
PORT_MASK	Port mask applied to the forwarding decision. MASK_MODE=4: PORT_MASK[52]: SRC_PORT_MASK_ENA. If set, SRC_PORT_MASK is AND'ed with destination result. PORT_MASK[51]: AGGR_PORT_MASK_ENA. If set, AGGR_PORT_MASK is AND'ed with destination result. MASK_MODE=4: PORT_MASK[50]: VLAN_PORT_MASK_ENA. If set, VLAN_PORT_MASK is AND'ed with destination result. PORT_MASK[7:0]: PGID table index.	55
MIRROR_PROBE	Mirroring performed according to configuration of a mirror probe. 0: No mirroring. 1: Mirror probe 0. 2: Mirror probe 1. 3: Mirror probe 2.	2

Table 85 • **VCAP IS2 Actions *(continued)***

Action Name	Description	Size
REW_CMD	REW_CMD[1:0] - PTP command: 0: No command. 1: ONE-STEP update. Update PTP PDU according to ingress and egress port mode. 2: TWO-STEP stamping. Send frame untouched, but store ingress and egress time stamps in the time stamp FIFO. 3: TIME-OF-DAY update. Fill system time-of-day into frame. REW_CMD[3:2] - Delay command: 0: No command. 1: Add configured egress delay for the egress port (REW::PTP_EDLY_CFG). 2: Add configured ingress delay for the ingress port (REW::PTP_IDLY1_CFG). 3: Add configured ingress delay for the ingress port (REW::PTP_IDLY2_CFG). REW_CMD[5:4] - Sequence number and time stamp command: 0: No command. 1: Fill in sequence number and increment. Sequence number is selected by IFH.TSTAMP. 2: Enable Delay_Req/resp processing. This mode includes updating the requestReceiptTimestamp with the ingress time stamp. 3: Enable Delay_Req/resp processing. This mode excludes updating the requestReceiptTimestamp. Note that if REW_CMD[5] is set, REW_CMD[5:4] is cleared by ANA_ACL before passing the REW_CMD action to the rewriter. REW_CMD[6]: If set, set DMAC to incoming frame's SMAC. REW_CMD[7]: If set, set SMAC to configured value for the egress port.	8
TTL_UPDATE_ENA	If set, IPv4 TTL or IPv6 hop limit is decremented. Overloading: If ACL_RT_MODE[3]=1 (SWAPPING) and ACL_RT_MODE[2:0]=4, 5, 6, or 7, the IPv4 TTL or IPv6 hop limit is preset to value configured in ANA_ACL::SWAP_IP_CTRL.	1
SAM_SEQ_ENA	If set, ACL_RT_MODE only applies to frames classified as SAM_SEQ. Overloading: If ACL_RT_MODE=0x8 (replace DMAC): If set, ANA_ACL::SWAP_IP_CTRL.DMAC_REPL_OFFSET_VAL controls number of bits in frame's DMAC, counting from MSB, to replace with corresponding bits in ACL_MAC.	1
TCP_UDP_ENA	If set, TCP/UDP ports are set to new values: DPORT = MATCH_ID, SPORT = MATCH_ID_MASK.	1
MATCH_ID	Logical ID for the entry. The MATCH_ID is extracted together with the frame if the frame is forwarded to the CPU (CPU_COPY_ENA). The result is placed in IFH.CL_RSLT. Overloading: If TCP_UDP_ENA is set, MATCH_ID defines new DPORT.	16
MATCH_ID_MASK	Mask used by MATCH_ID. Overloading: If TCP_UDP_ENA is set, MATCH_ID defines new SPORT.	16
CNT_ID	Counter ID, used per lookup to index the 4K frame counters (ANA_ACL:CNT_TBL). Multiple VCAP IS2 entries can use the same counter.	12
SWAP_MAC_ENA	Swap MAC addresses. Overloading: If REW_CMD[5]=1: For multicast Delay_Resp frames, SWAP_MAC_ENA and LOG_MSG_INT encodes a new logMessageInterval. See LOG_MSG_INT.	1

Table 85 • **VCAP IS2 Actions *(continued)***

Action Name	Description	Size
ACL_RT_MODE	Controls routing updates in ANA_AC and how ACL_MAC is applied: 0: No change. ACL_RT_MODE[3] = 0 (ROUTING mode): ACL_RT_MODE[0] = 1: Enable Unicast routing updates in ANA_AC ACL_RT_MODE[1] = 1: Enable Multicast routing updates in ANA_AC ACL_RT_MODE[2] = 1: Enable overloading of ACL_MAC field. ACL_RT_MODE[3] = 1 (SWAPPING mode): ACL_RT_MODE[2:0] encodes a SWAPPING sub-mode: 0: Replace DMAC. 1: Replace SMAC. 2: Swap MAC addresses. Replace SMAC if new SMAC is multicast. 3: Swap MAC addresses if DMAC is unicast. Replace SMAC if DMAC is multicast. 4: Swap MAC addresses. Replace SMAC if new SMAC is multicast. Swap IP addresses. Replace SIP if new SIP is multicast. 5: Swap MAC addresses if DMAC is unicast. Replace SMAC if DMAC is multicast. Swap IP addresses if DIP is unicast. Replace SIP if DIP is multicast. 6: Swap IP addresses. Replace SIP if new SIP is multicast. 7: Replace SMAC. Swap IP addresses. Replace SIP if new SIP is multicast. Note that when replacing a MAC address, the new MAC address is taken from ACL_MAC. When replacing a SIP, the new SIP is taken from ANA_ACL::SWAP_SIP[POLICE_IDX].	4
ACL_MAC	MAC address used to replace either SMAC or DMAC based on ACL_RT_MODE. Overloading: ACL_RT_MODE[3:2] = 1. ACL_MAC[0]: PORT_PT_CTRL_ENA - Overrule PORT_PT_CTRL. ACL_MAC[6:1]: PORT_PT_CTRL_IDX - into ANA_AC_POL:PORT_PT_CTRL. ACL_MAC[16]: Enable use of ACL_MAC[26:17] as DLB policer index. ACL_MAC[26:17]: DLB policer index. ACL_RT_MODE(3) = 1: MAC address used to replace either SMAC or DMAC based on ACL_RT_MODE.	48
PTP_DOM_SEL	PTP domain selector. PTP_DOM_SEL indexes the PTP configuration in ANA_ACL:PTP_DOM used in Delay_Resp frames. See REW_CMD.	2

3.19 Analyzer Access Control Lists

The analyzer access control list (ANA_ACL) block performs the following tasks.

- Controls the VCAP IS2 lookups in terms of key selection, evaluation of range checkers, and control of specific VCAP IS2 key fields such as IGR_PORT_MASK_SEL and VID.
- Generates VCAP IS2 keys and execute the two lookups.
- Updates VCAP IS2 statistics based on match results.
- Assigns actions from VCAP IS2 matching or default actions when there is no match.
- Controls routing-related frame rewrites.

The following sections provides specific information about each of these tasks.

3.19.1 VCAP IS2

Each frame is classified to one of nine overall VCAP IS2 frame types. The frame type determines which VCAP IS2 key types are applicable and controls which frame data to extract.

Table 86 • **VCAP IS2 Frame Types**

Frame Type	Condition
IPv6 multicast frame	The Type/Len field is equal to 0x86DD. The IP version is 6. The destination IP address is a multicast address (FF00::/8). Special IPv6 frames: •IPv6 TCP frame: Next header is TCP (0x6). •IPv6 UDP frame: Next header is UDP (0x11). •IPv6 Other frame: Next header is neither TCP nor UDP.
IPv6 unicast frame	The Type/Len field is equal to 0x86DD. The IP version is 6. The destination IP address is not a multicast address. Special IPv6 frames: •IPv6 TCP frame: Next header is TCP (0x6). •IPv6 UDP frame: Next header is UDP (0x11). •IPv6 Other frame: Next header is neither TCP nor UDP
IPv4 multicast frame	The Type/Len field is equal to 0x800. The IP version is 4. The destination IP address is a multicast address (224.0.0.0/4). Special IPv4 frames: •IPv4 TCP frame: IP protocol is TCP (0x6). •IPv4 UDP frame: IP protocol is UDP (0x11). •IPv4 Other frame: IP protocol is neither TCP nor UDP.
IPv4 unicast frame	The Type/Len field is equal to 0x800. The IP version is 4. The destination IP address is not a multicast address. Special IPv4 frames: •IPv4 TCP frame: IP protocol is TCP (0x6). •IPv4 UDP frame: IP protocol is UDP (0x11). •IPv4 Other frame: IP protocol is neither TCP nor UDP.
(R)ARP frame	The Type/Len field is equal to 0x0806 (ARP) or 0x8035 (RARP).
OAM Y.1731 frame	The Type/Len field is equal to 0x8902 (ITU-T Y.1731).
SNAP frame	The Type/Len field is less than 0x600. The Destination Service Access Point field, DSAP is equal to 0xAA. The Source Service Access Point field, SSAP is equal to 0xAA. The Control field is equal to 0x3.
LLC frame	The Type/Len field is less than 0x600 The LLC header does not indicate a SNAP frame.
ETYPE frame	The Type/Len field is greater than or equal to 0x600, and the Type field does not indicate any of the previously mentioned frame types, that is, ARP, RARP, OAM, IPv4, or IPv6.

Some protocols of interest do not have their own frame type but are included in other mentioned frame types. For instance, Precision Time Protocol (PTP) frames are handled by VCAP IS2 as either ETYPE frames, IPv4 frames, or IPv6 frames. The following encapsulations of PTP frames are supported:

- PTP over Ethernet: ETYPE frame with Type/Len = 0x88F7.
- PTP over UDP over IPv4: IPv4 UDP frame with UDP destination port numbers 319 or 320.
- PTP over UDP over IPv6: IPv6 UDP frame with UDP destination port numbers 319 or 320.

Specific PTP fields are contained in eight bytes of payload extracted after either the EtherType for PTP over Ethernet or in eight bytes of payload after the UDP header for IPv4/IPv6 frames.

3.19.1.1 Port Configuration and Key Selection

This section provides information about special port configurations that control the key generation for VCAP IS2.

The following table lists the registers associated with port configuration for VCAP IS2.

Table 87 • **Port Configuration of VCAP IS2**

Register	Description	Replication
ANA_ACL:PORT:VCAP_S2_KEY_SEL	Configuration of the key selection for the VCAP IS2	Per port per lookup
ANA_ACL::VCAP_S2_CFG	Enabling of VCAP IS2 lookups.	Per port per lookup

Each of the two lookups in VCAP IS2 must be enabled before use (VCAP_S2_CFG.SEC_ENA) for a key to be generated and a match to be used. If a lookup is disabled on a port, frames received by the port are not matched against rules in VCAP IS2 for that lookup.

Each port controls the key selection for each of the two lookups in VCAP IS2 through the VCAP_S2_KEY_SEL register. For some frame type (OAM Y.1731, SNAP, LLC, EYTYPE) the key selection is given as only the MAC_ETYPE key can be used. For others frames types (ARP and IP) multiple keys can be selected from. The following table lists the applicable key types available for each frame type listed in Table 86, page 167.

Table 88 • **VCAP IS2 Key Selection**

Frame Type	Applicable VCAP IS2 keys
IPv6 multicast frames	IP6_VID IP_7TUPLE MAC_ETYPE
IPv6 unicast frames	IP_7TUPLE MAC_ETYPE
IPv4 multicast frames	IP4_VID IP_7TUPLE IP4_TCP_UDP IP4_OTHER MAC_ETYPE
IPv4 unicast frames	IP_7TUPLE IP4_TCP_UDP IP4_OTHER MAC_ETYPE
(R)ARP frames	ARP MAC_ETYPE
OAM Y.1731 frames	MAC_ETYPE
SNAP frames	MAC_ETYPE
LLC frames	MAC_ETYPE
ETYPE frames	MAC_ETYPE

Note: The key selection is overruled if VCAP CLM instructs the use of a custom key, where data from the frame is extracted at a VCAP CLM-selected point. For more information, see VCAP IS2 Custom Keys, page 169.

For information about VCAP IS2 keys and actions, see VCAP IS2 Keys and Actions, page 156.

3.19.1.2 Key Selection for Non-ETH Frames

VCAP IS2 also support IP over MPLS over Ethernet frames of frame_type CW (IFH.TYPE_AFTER_POP). For such frames, the VCAP IS2 keys are generated with data starting at the IP header after the MPLS label stack. Link layer information is not available. Keys which contain link layer fields (for instance field L2_DMAC) have these fields nulled.

The key selection is based on information in the first nibble of the IP header (control word). If the nibble equals 6, the key selection for IPv6 frames is used. If the nibble equals 4, the key selection for IPv4 frame is used. For other values of the first nibble, no keys are generated. VCAP IS2 custom key must be used instead.

3.19.1.3 VCAP IS2 Custom Keys

VCAP IS2 supports two custom keys, CUSTOM_1 and CUSTOM_2. Custom keys extract raw data from the frame at one selectable location. The custom keys enable matching against protocol fields not supported by other VCAP IS2 keys. The use of the custom keys must be enabled through a VCAP CLM FULL action using CUSTOM_ACE_ENA and CUSTOM_ACE_OFFSET.

CUSTOM_1 extracts 57 consecutive bytes of data from the frame and CUSTOM_2 extracts 24 consecutive bytes of data from the frame. The data is extracted at one of four different positions in the frame. For frames of frame_type ETH (IFH.TYPE_AFTER_POP), the positions are defined as:

- Link layer. Corresponds to first byte in DMAC.
- Link layer payload. Corresponds to first byte in L3, after VLAN tags and EtherType.
- Link layer payload plus 20 bytes. Corresponds to first byte in L4 for IPv4 frame.
- Link layer payload plus 40 bytes. Corresponds to first byte in L4 for IPv6 frame.

For frames of frame_type CW, the first link layer position is not available. The positions are defined as:

- Link layer payload. Corresponds to first byte in control word for non-IP frames and first byte in IP header for IP frames.
- Link layer payload plus 20 bytes. Corresponds to first byte in L4 for IPv4 frame.
- Link layer payload plus 40 bytes. Corresponds to first byte in L4 for IPv6 frame.

3.19.1.4 VCAP IS2 Range Checkers

The following table lists the registers associated with configuring VCAP IS2 range checkers.

Table 89 • **VCAP IS2 Range Checker Configuration**

Register	Description	Replication
ANA_ACL::VCAP_S2_RNG_CTRL	Configuration of the range checker types	Per range checker
ANA_ACL::VCAP_S2_RNG_VALUE_CFG	Configuration of range start and end points	Per range checker
ANA_ACL::VCAP_S2_RNG_OFFSET_CFG	Offset into frame when using a range checker based on selected value from frame	None

Eight global VCAP IS2 range checkers are supported by the IP4_TCP_UDP and IP_7TUPLE keys. All frames using these keys are compared against the range checkers and a 1-bit range “match/no match” flag is returned for each range checker. The combined eight match/no match flags are used in the L4_RNG key field. So, it is possible to include for example ranges of DSCP values and/or ranges of TCP source port numbers in the ACLs.

Note, VCAP CLM also supports range checkers but they are independent of the VCAP IS2 range checkers.

The range key is generated for each frame based on extracted frame data and the configuration in ANA_ACL::VCAP_S2_RNG_CTRL. Each of the eight range checkers can be configured to one of the following range types:

- TCP/UDP destination port range
  Input to the range is the frame’s TCP/UDP destination port number.
  Range is only applicable to TCP/UDP frames.

- TCP/UDP source port range
  Input to the range is the frame's TCP/UDP source port number.
  Range is only applicable to TCP/UDP frames.
- TCP/UDP source and destination ports range. Range is matched if either source or destination port is within range.
  Input to the range are the frame's TCP/UDP source and destination port numbers.
  Range is only applicable to TCP/UDP frames.
- VID range
  Input to the range is the classified VID.
  Range is applicable to all frames.
- DSCP range
  Input to the range is the classified DSCP value.
  Range is applicable to IPv4 and IPv6 frames.
- Selected frame field range
  Input to the range is a selected 16-bit wide field from the frame. The field is selected using the offset value configured in ANA_ACL::VCAP_S2_RNG_OFFSET_CFG. The offset starts at the Type/Len field in the frame after up to 3 VLAN tags have been skipped. Note that only one selected value can be configured for all eight range checkers.
  Range is applicable to all frames.

Range start points and range end points are configured in ANA_ACL::VCAP_S2_RNG_VALUE_CFG with both the start point and the end point being included in the range. A range matches if the input value to the range (for instance the frame's TCP/UDP destination port) is within the range defined by the start point and the end point.

3.19.1.5 Miscellaneous VCAP IS2 Key Configurations

The following table lists the registers associated with configuration that control the specific fields in the VCAP IS2 keys.

Table 90 • **Other VCAP IS2 Key Configurations**

Register	Description	Replication
ANA_ACL::VCAP_S2_CFG.SEC_ROUTE_HANDLING_ENA	Enables use of IRLEG VID and ERLEG VID instead of classified VID.	Per port
ANA_ACL:VCAP_S2:VCAP_S2_MISC_CTRL	Configuration of IGR_PORT_MASK_SEL and IGR_PORT_MASK.	None

By default, the field VID contains the frame's classified VID for both VCAP IS2 lookups. Routable frames (determined by ANA_L3) with field L3_RT=1, can be configured (ANA_ACL::VCAP_S2_CFG.SEC_ROUTE_HANDLING_ENA) to use the IRLEG VID for the first lookup in VCAP IS2 and ERLEG VID for the second lookup. This enables ACL rules that apply to the ingress router leg and ACL rules that apply to the egress router leg. The IRLEG VID and ERLEG VID are provided by ANA_L3.

All VCAP IS2 keys except IP4_VID and IP6_VID contain the field IGR_PORT_MASK_SEL and IGR_PORT_MASK. For frames received on a front port, IGR_PORT_MASK_SEL is set to 0 and the bit corresponding to the frame's physical ingress port number is set in IGR_PORT_MASK. The following lists some settings for frames received on other interfaces and how this can be configured through register VCAP_S2_MISC_CTRL:

- **Loopback frames.** By default, IGR_PORT_MASK_SEL = 1 and IGR_PORT_MASK has one bit set corresponding to the physical port which the frame was looped on. Optionally use IGR_PORT_MASK_SEL = 3.
- **Masqueraded frames.** By default, IGR_PORT_MASK_SEL=0 and IGR_PORT_MASK has one bit set corresponding to the masquerade port. Optionally, use IGR_PORT_MASK_SEL = 2.
- **Virtual device frames.** By default, IGR_PORT_MASK_SEL=0 and IGR_PORT_MASK has one bit set corresponding to the original physical ingress port. Optionally use IGR_PORT_MASK_SEL = 3.
- **CPU injected frames.** By default, IGR_PORT_MASK_SEL = 0 and IGR_PORT_MASK has none bits set. Optionally use IGR_PORT_MASK_SEL = 3.

For more information about the contents of IGR_PORT_MASK_SEL and IGR_PORT_MASK, see VCAP IS2 Key Overview, page 157.

3.19.1.6 Processing of VCAP IS2 Actions

VCAP IS2 returns an action for each enabled VCAP IS2 lookup. If the lookup matches an entry in VCAP IS2 then the associated action is returned. A default action is returned when no entries are matched. The first VCAP IS2 lookup returns a default action per physical ingress port while the second VCAP IS2 lookup returns a common default action. There is no difference between an action from a match and a default action.

VCAP IS2 contains 58 default actions that are allocated the following way:

- 0-52: Per ingress port default actions for first lookup.
- 53, 54: Per CPU port (CPU0 and CPU1) default actions for first lookup.
- 55, 56: Per virtual device (VD0 and VD1) default actions for first lookup.
- 57: Common default action for second lookup.

Each of the two VCAP IS2 actions (from first and second lookups) can be processed either at the pipeline point ANA_OU_SW or at pipeline point ANA_IN_SW. This is controlled by VCAP IS2 action IS_INNER_ACL. The order of processing is given as:

1. Process first action if the first's action IS_INNER_ACL is 0.
2. Process second action if the second's action IS_INNER_ACL is 0.
3. Process first action if the first's action IS_INNER_ACL is 1.
4. Process second action if the second's action IS_INNER_ACL is 1.

This implies that the second action can be processed in ANA_OU_SW pipeline point and the first action can be processed in ANA_IN_SW pipeline point, or vice versa. Both actions can also be placed at the same pipeline point in which case the first action is processed before the second.

Processing of each action obeys the frame's current pipeline information (pipeline point and pipeline action). For instance if the frame has been redirected at pipeline point ANA_CLM then neither of the VCAP IS2 actions are applied. However, VCAP IS2 action IGNORE_PIPELINE_CTRL can overrule this.

Furthermore, frame processing between pipeline points ANA_OU_SW and ANA_IN_SW such as OAM filtering or protection switching can influence the processing of the VCAP IS2 actions so that only an action belonging to pipeline point ANA_OU_SW is processed.

The following table lists how the two VCAP IS2 action are combined when both VCAP IS2 actions are processed. For most cases, the processing is sequential, meaning that if both actions have settings for the same (for instance MIRROR_PROBE), the second processing takes precedence. Others are sticky meaning they cannot be undone by the second processing if already set by the first.

Table 91 • **Combining VCAP IS2 Actions**

Action name	Combining actions
IS_INNER_ACL	Processed individually for each action.
PIPELINE_FORCE_ENA	Processed individually for each action. Second processing uses the result from the first processing which can prevent its processing if the pipeline information from first processing disables the second processing.
PIPELINE_PT	See PIPELINE_FORCE_ENA.
HIT_ME_ONCE	Sticky.
INTR_ENA	Sticky.
CPU_COPY_ENA	Sticky.
CPU_QU_NUM	Sticky (each action can generate a CPU copy and based on configuration in ANA_AC:PS_COMMON:CPU_CFG.ONE_CPU_COPY_ONLY_MASK, both copies can be sent to CPU).
CPU_DIS	Processed individually for each action. Second processing may clear CPU copy from first processing.

Table 91 • **Combining VCAP IS2 Actions *(continued)***

Action name	Combining actions
LRN_DIS	Sticky.
RT_DIS	Sticky.
POLICE_ENA	Sticky.
POLICE_IDX	Second action takes precedence when POLICE_ENA is set.
IGNORE_PIPELINE_CTRL	Processed individually for each action.
LOG_MSG_INT	Second action takes precedence when LOG_MSG_INT > 0.
MASK_MODE	Processed individually for each action. Second processing uses the result from the first processing. (If the first processing is sticky, the second processing is not applied). MASK_MODE is sticky for the following values: 3: REPLACE_ALL 4: REDIR_PGID 6: VSTAX
PORT_MASK	See MASK_MODE.
MIRROR_PROBE	Second action takes precedence when MIRROR_PROBE > 0.
REW_CMD	Second action takes precedence same functions are triggered by both actions. If different functions are used, both are applied.
TTL_UPDATE_ENA	Second action takes precedence when same function is triggered by both actions. If different functions are used, both are applied.
SAM_SEQ_ENA	Second action takes precedence when same function is triggered by both actions. If different functions are used, both are applied.
TCP_UDP_ENA	Second action takes precedence when TCP_UDP_ENA is set.
MATCH_ID	Processed individually for each action. Second processing uses the result from the first processing.
MATCH_ID_MASK	Processed individually for each action. Second processing uses the result from the first processing.
CNT_ID	Processed individually for each action. Two counters are updated, one for each action.
SWAP_MAC_ENA	Sticky.
ACL_RT_MODE	Second action takes precedence when same function is triggered by both actions. If different functions are used, both are applied.
ACL_MAC	Second action takes precedence when same function is triggered by both actions. If different functions are used, both are applied.
PTP_DOM_SEL	Second action takes precedence when PTP_DOM_SEL > 0.

3.19.1.7 VCAP IS2 Statistics and Diagnostics

The following table lists the registers associated with VCAP IS2 statistics and diagnostics.

Table 92 • **VCAP IS2 Statistics and Diagnostics**

Register	Description	Replication
ANA_ACL::SEC_LOOKUP_STICKY	Sticky bits for each key type used in VCAP IS2.	Per lookup
ANA_ACL::CNT	VCAP IS2 match counters. Indexed with VCAP IS2 action CNT_ID.	1,024

Each key type use in a lookup in VCAP IS2 triggers a sticky bit being set in ANA_ACL::SEC_LOOKUP_STICKY. A sticky bit is cleared by writing.

Each action returned by VCAP IS2 triggers one of 1,024 counters (ANA_ACL::CNT) to be incremented. This applies to both actions from matches and default actions. The index into the counters is provided by VCAP IS2 action CNT_ID. The CNT_ID is fully flexible meaning multiple actions can point to the same counter. The counters are writable so they can be preset to any value. A counter is cleared by writing 0.

3.19.1.8 Routing Statistics

If a routed frame is discarded by a VCAP IS2 rule it is configurable how the frame is counted in terms of ingress and egress router leg statistics.

If the frame is discarded by the first VCAP IS2 lookup, it is selectable whether to count the frame in the ingress router leg statistics (ANA_ACL::VCAP_S2_MISC_CTRL.ACL_RT_IGR_RLEG_STAT_MODE). The frame is never counted by the egress router leg statistics.

If the frame is discarded by the second VCAP IS2 lookup, it is selectable whether to count the frame in the egress router leg statistics (ANA_ACL::VCAP_S2_MISC_CTRL.ACL_RT_EGR_RLEG_STAT_MODE). The frame is always counted by the ingress router leg statistics.

3.19.2 Analyzer Access Control List Frame Rewriting

The ANA_ACL block supports various frame rewrite functions used in routing, PTP, and OAM.

When rewriting in the ANA_ACL block, ingress mirroring with an exact mirror copy of the incoming frame is no longer possible, because any rewrites are also reflected in the mirror copy.

3.19.2.1 Address Swapping and Rewriting

VCAP IS2 actions can trigger frame rewrites in the ANA_ACL block with focus on MAC addresses, IP addresses, TCP/UDP ports as well as IPv4 TTL and IPv6 hop limit.

The following table lists the registers associated with address swapping and rewriting capabilities in ANA_ACL.

Table 93 • **ANA_ACL Address Swapping and Rewriting Registers**

Register	Description	Replication
ANA_ACL::SWAP_SIP	IP table with 32 IPv4 addresses or 8 IPv6 addresses. Provides new source IP address when swapping IP addresses in frame with multicast destination IP address.	32
ANA_ACL::SWAP_IP_CTRL	Controls IPv4 TTL and IPv6 hop limit values. Controls number of bits to replace in DMAC.	None

VCAP IS2 action ACL_RT_MODE can enable the following rewrite modes:

- **Replace DMAC.** The frame's DMAC is replaced with the MAC address specified in VCAP IS2 action ACL_MAC. It is selectable to only replace part of the DMAC (for instance the OUI only). This is enabled with VCAP IS2 action REW_CMD[9], and the number of bits to replace is set in ANA_ACL::SWAP_IP_CTRL.DMAC_REPL_OFFSET_VAL.
- **Replace SMAC.** The frame's SMAC is replaced with the MAC address specified in VCAP IS2 action ACL_MAC.
- **Swap MAC addresses and replace SMAC if MC.** The frame's MAC addresses are swapped. If the new SMAC is a multicast MAC, it is replaced with the MAC address specified in VCAP IS2 action ACL_MAC.
- **Swap MAC addresses if UC else replace SMAC if MC.** The frame's MAC addresses are swapped if the frame's DMAC is unicast. Otherwise, the frame's SMAC is replaced with the MAC address specified in VCAP IS2 action ACL_MAC.
- **Swap MAC and IP addresses and replace SIP and SMAC if MC.** The frame's MAC addresses are swapped and the frame's IP addresses are swapped. If the new SMAC is a multicast MAC, it is replaced with the MAC address specified in VCAP IS2 action ACL_MAC. If the new SIP is a multicast, it is replaced with an IP address from the IP address table ANA_ACL::SWAP_SIP.

- **Swap MAC and IP addresses if UC else replace SIP and SMAC if MC.** The frame's MAC addresses are swapped if the frame's DMAC is unicast. Otherwise, the frame's SMAC is replaced with the MAC address specificed in VCAP IS2 action ACL_MAC. The frame's IP addresses are swapped if the frame's DIP is unicast. Otherwise, the frame's SIP is replaced with an IP address from the IP address table ANA_ACL::SWAP_SIP.
- **Swap IP addresses and replace SIP if MC.** The frame's IP addresses are swapped. If the new SIP is a multicast, it is replaced with an IP address from the IP address table ANA_ACL::SWAP_SIP.
- **Replace SMAC, swap IP addresses, and replace SIP if MC.** The frame's SMAC is replaced with the MAC address specificed in VCAP IS2 action ACL_MAC. The frame's IP addresses are swapped. If the new SIP is a multicast, it is replaced with an IP address from the IP address table ANA_ACL::SWAP_SIP.

The IP address table (ANA_ACL::SWAP_SIP) used by some of the above mentioned modes is indexed using VCAP IS2 action POLICE_IDX (VCAP IS2 action POLICE_ENA must be 0). The IP address table can contain 32 IPv4 addresses or 8 IPv6 addresses.

When swapping IP addresses or replacing the source IP address, it is also possible to preset the IPv4 TTL or the IPv6 hop limit. This is enabled by VCAP IS2 action TTL_UPDATE_ENA. The new TTL value for IPv4 frames is configured in ANA_ACL::SWAP_IP_CTRL.IP_SWAP_IP4_TTL_VAL and the new hop limit value for IPv6 frames is configured in ANA_ACL::SWAP_IP_CTRL.IP_SWAP_IP4_TTL_VAL.

IPv4 and IPv6 TCP/UDP frames have the option to replace the source and destination port numbers with new values. This is enabled through VCAP IS2 action TCP_UDP_ENA and new port numbers are provided by VCAP IS2 action MATCH_ID and MATCH_ID_MASK.

The IPv4 header checksum is updated whenever required by above mentioned frame rewrites. The UDP checksum is updated whenever required for IPv6 frames and cleared for IPv4 frames.

3.19.2.2 Routing in ANA_ACL

The following table lists the registers associated with routing capabilities in ANA_ACL.

Table 94 • **ANA_ACL Routing Registers**

Register	Description	Replication
ANA_ACL::VCAP_S2_MISC_CTRL.ACL_RT_SEL	Enable routing related frame rewrites in ANA_ACL	None
ANA_ACL::VCAP_S2_MISC_CTRL.ACL_RT_UPDATE_GEN_IDX_ERLEG_ENA	Enable use of egress router leg as VSI in rewriter.	None
ANA_ACL::VCAP_S2_MISC_CTRL.ACL_RT_UPDATE_GEN_IDX_EVID_ENA	Enable use of egress VID as VSI in rewriter.	None

If PTP and routing is to be supported concurrently, then some routing related frame rewrites must be done in ANA_ACL instead of in the rewriter. This is enabled in ANA_ACL::VCAP_S2_MISC_CTRL.ACL_RT_SEL or by setting VCAP IS2 action ACL_RT_MODE. When enabled, the frame's DMAC is changed to the next-hop MAC address specificed by ANA_L3 and the rewriter is informed not to rewrite the DMAC. The following classification results used by the rewriter are changed when routing in ANA_ACL:

- The frame's classified VID is set to the egress VID.
- The frame's classified VSI is optionally set to the egress router leg (ANA_ACL::VCAP_S2_MISC_CTRL.ACL_RT_UPDATE_GEN_IDX_ERLEG_ENA) or the egress VID (ANA_ACL::VCAP_S2_MISC_CTRL.ACL_RT_UPDATE_GEN_IDX_EVID_ENA)

In addition, ANA_L3 must be setup to change the source MAC address with the address belonging to the egress router leg (ANA_L3::ROUTING_CFG.RT_SMAC_UPDATE_ENA). The IPv4 TTL or IPv6 hop limit is still decremented by the rewriter.

3.19.2.3 PTP Delay_Req/Resp Processing

VCAP IS2 actions can trigger PTP Delay_Req/resp processing where an incoming Delay_Req frame is terminated and a Delay_Resp frame is generated with appropriate PTP updates.

The following table lists the registers associated with PTP processing and rewriting capabilities in ANA_ACL.

Table 95 • **ANA_ACL PTP Processing and Rewriting Registers**

Register	Description	Replication
ANA_ACL:PORT:PTP_CFG	PTP time domain configuration and PTP portNumber configuration used in portIdentity.	Per port
ANA_ACL::PTP_MISC_CTRL	Enable Delay_Req/resp processing. Enable logMessageInterval update for multicast Delay_Resp frames.	None
ANA_ACL:PTP_DOM:PTP_CLOCK_ID_MSB	PTP clockIdentity used in portIdentity.	Per PTP domain
ANA_ACL:PTP_DOM:PTP_CLOCK_ID_LSB	PTP clockIdentity used in portIdentity.	Per PTP domain
ANA_ACL:PTP_DOM:PTP_SRC_PORT_CFG	PTP portNumber used in portIdentity. Configuration of whether to use fixed portNumber or per-port portNumber.	Per PTP domain
ANA_ACL:PTP_DOM:PTP_MISC_CFG	Configuration of flagField with associated mask. The PTP domain is specified with VCAP IS2 action PTP_DOM_SEL.	Per PTP domain

VCAP IS2 action REW_CMD defines the Delay_Req/Resp processing mode where two modes exist:

- Generate Delay_Resp from Delay Req frame and update the receiveTimestamp in the PTP header with the ingress PTP time stamp. Use the time stamp from the configured time domain (ANA_ACL:PORT:PTP_CFG.PTP_DOMAIN). This time domain must match the selected time domain on the ingress port (DEVxx::PTP_CFG.PTP_DOM)
- Generate Delay_Resp from Delay Req frame and do not update the receiveTimestamp.

In addition to the VCAP IS2 action, the Delay_Req/Resp processing must also be enabled in ANA_ACL (ANA_ACL::PTP_MISC_CTRL.PTP_ALLOW_ACL_REW_ENA).

In addition to updating the receiveTimestamp, the following PTP header modifications are done independently of the Delay_Req/Resp processing mode:

- **messageType.** The PTP messageType is set to 0x9 (Delay_Resp)
- messageLength. The PTP messageLength is set to 0x54 and the frame is expanded with 10 bytes to accomodate the requestingPortIdentity. If the frame expands beyond 148 bytes, then the frame is discarded.

  For PTP frames carried over UDP, the UDP length is increased with 10 bytes. It is mandatory that the original UDP length is 44 bytes. For PTP frames carried over IPv4, the IP total length is increased with 10 bytes.
- **flagField.** Selected bits in the PTP flagField can be reconfigured using the per-PTP domain configured flagField and flagField mask (ANA_ACL:PTP_DOM:PTP_MISC_CFG).

  The PTP domain is selected using VCAP IS2 action PTP_DOM_SEL.
- **sourcePortIdentity.** The sourcePortIdentity is constructed using a 64-bit clockIdentity and a 16-bit portNumber. The clockIdentity is configurable per-PTP domain.

  By default, the portNumber is set to a per-PTP domain configured value. In addition, it is selectable to overwrite the six least significant bits with a per-port configured value.

  The PTP domain is selected using VCAP IS2 action PTP_DOM_SEL.
- **controlField.** The PTP controlField is set to 0x3.
- **logMessageInterval.** The PTP logMessageInterval can be specified for multicast frames only with new values ranging from –8 through 7. The new value is configured using VCAP IS2 actions SWAP_MAC_ENA and LOG_MSG_INT.

This update can be globally disabled in ANA_ACL::PTP_MISC_CTRL.PTP_DELAY_REQ_MC_UPD_ENA.

- **requestingPortIdentity.** This fields is set to the incoming frame's sourcePortIdentity.

For PTP frames carried over UDP, any PTP header modifications trigger the UDP checksum to be updated for IPv6 frames and to be cleared for IPv4 frames.

In addition to the PTP header modifications, address swapping and rewriting can also be enabled for PTP frames. For PTP carried over UDP it can be useful to also swap MAC addresses, swap IP addresses, change UDP port numbers, and preset the IPv4 TTL or IPv6 hop limit.

3.20 Analyzer Layer 2 Forwarding and Learning

The analyzer Layer 2 (ANA_L2) block performs the following tasks:

- Determining forwarding decisions
- Tracking network stations and their MAC addresses through learning and aging
- Tracking and setting limits for number station learned per FID/ports
- Scanning and updating the MAC table

The analyzer Layer 2 block makes a forwarding decision based on a destination lookup in the MAC table. The lookup is based on the received Destination MAC address together with either the classified VID for multicast traffic or the classified FID for unicast traffic. If an entry is found in the MAC table lookup, the associated entry address is used for selecting forwarding port or ports. A flood forwarding decision is made if no entry is found and forwarding to unknown destinations is permitted (ANA_L3:VLAN:VLAN_CFG.VLAN_SEC_FWD_ENA).

The analyzer Layer 2 block performs a source lookup in the MAC table to determine if the source station is known or unknown by looking at if there is an entry in the MAC table. If a MAC table entry exists, a port move detection is performed by looking at whether the frame was received at the expected interface specified. The source check can trigger the following associated actions.

- Disabling forwarding from unknown or moved stations
- Triggering automated learning
- Sending copy to CPU

If the source station is known, the entry AGE_FLAG is cleared to indicate that source associated with the entry is active. The AGE_FLAG is part of the aging functionality to remove inactive MAC table entries. For more information, see Automated Aging (AUTOAGE), page 189.

The following subsections further describe the main analyzer Layer 2 blocks.

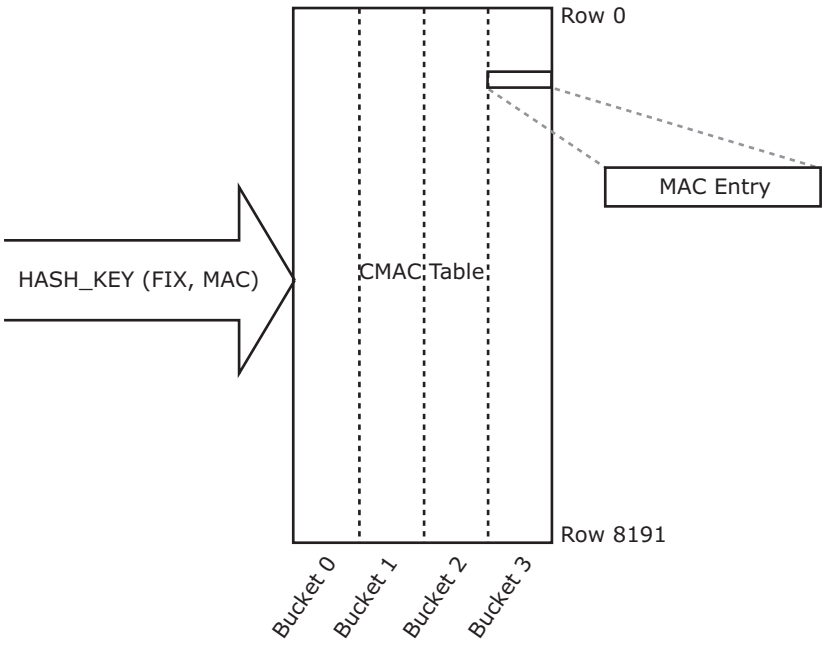
3.20.1 Analyzer MAC Table

The analyzer Layer 2 block contains a MAC table with 32,768 entries containing information about stations learned or known by the device. The table is organized as a hash table with four buckets on each row. Each row is indexed based on a hash value calculated based on the station's (FID, MAC) pair.

The filtering ID (FID) used is either the FID from the VLAN table if a unicast MAC is looked up or else it is the classified VID. It is possible to enforce use of FID for multicast (ANA_L3:COMMON:SERVICE_CFG).

MAC table organization is depicted in the following illustration.

Figure 46 • **MAC Table Organization**



3.20.1.1 MAC Entry Table

Each entry in the MAC table contains the fields listed in the following table.

Table 96 • **MAC Table Entry**

Field	Bits	Description
VLD	1	Entry is valid.
MAC	48	The MAC address of the station. (Part of primary key).
FID	13	VLAN filtering identifier (FID) unicast C-MAC entries. VLAN identifier (VID) for multicast C-MAC entries. (Part of primary key).
LOCKED	1	Lock the entry as static. Note: Locking an entry will prevent hardware from changing anything but the AGE_FLAG in the entry.
MIRROR	1	Frames from or to this station are candidate for mirroring.
AGE_FLAG	2	The number of loopbackaging periods that have taken place since the last frame was received from this station. Incremented by hardware during a CPU SCAN and AUTOAGE SCAN for the entry configured AGE_INTERVAL.
AGE_INTERVAL	2	Used to select which age timer is associated with the entry.
NXT_LRN_ALL	1	Used to ensure consistent MAC tables in a stack.
CPU_COPY	1	Frames from or to this station are candidate for CPU copy. Used together with CPU_QU (to specify queue number).
CPU_QU	3	Used together with CPU_COPY.
SRC_KILL_FWD	1	Frames from this station will not be forwarded. This flag is not used for destination lookups.
VLAN_IGNORE	1	Used for ignoring the VLAN_PORT_MASK contribution from the VLAN table when forwarding frames to this station. Can optionally be used for source port ignore (ANA_L2::FWD_CFG.FILTER_MODE_SEL).
ADDR	12	Stored entry address used for forwarding and port move detection. The field is encoded according to ADDR_TYPE.

Table 96 • **MAC Table Entry *(continued)***

Field	Bits	Description
ADDR_TYPE	3	This field encodes how to interpret the ADDR field. Encoded as: ADDR_TYPE= UPSID_PN(0): Unicast entry address. Used to associate the entry with a unique {UPSID, UPSPN} port. ADDR(9:5) = UPSID ADDR(4:0) = UPSPN ADDR_TYPE=GCPU_UPS(1): CPU management entry address. Used to associate the entry with a unique global CPU port or internal port. ADDR(9:5) = UPSID ADDR(11) = 0: ADDR(3:0) = CPU port number. ADDR(11) = 1: ADDR(3:0) = 0xe: Internal port number. ADDR(3:0) = 0xf: Local lookup at destination unit (UPSID). ADDR_TYPE=GLAG(2): Unicast GLAG address. Used to associate the entry with a global aggregated port. ADDR = GLAGID. ADDR_TYPE=MC_IDX(3): Multicast entry address. Used to associate the entry with an entry in the PGID table. Specifies forwarding to the ports found in the PGID table entry indexed by mc_idx. ADDR = mc_idx.

3.20.1.2 CAM Row

Under some circumstances, it might be necessary to store more entries than possible on a MAC table row, in which case entries are continuously changed or automatically updated.

To reduce this issue, one additional MAC table row (named CAM row) exist with entries that can be used to extend any hash chain, and thereby reduce the probability for hash depletion where all the buckets on a MAC table row are used. These entries on the CAM row can be used as any other MAC table entries.

CAM row usage is controlled separately (by ANA_L2::LRN_CFG.AUTO_LRN_USE_MAC_CAM_ENA, ANA_L2::LRN_CFG.CPU_LRN_USE_MAC_CAM_ENA and ANA_L2::LRN_CFG.LRN_MOVE_CAM_ENTRY_BACK).

3.20.2 MAC Table Updates

Entries in the MAC table are added, deleted, or updated by the following methods.

- Automatic (hardware-based) learning of source MAC addresses; that is, inserting new (MAC, FID) pairs in the MAC table
- CPU access to MAC table (for example, CPU-based learning and table maintenance)
- Automated age scan.

3.20.3 CPU Access to MAC Table

This section describes CPU access to the MAC table. The following table lists the applicable registers.

Table 97 • **MAC Table Access Registers**

Target::Register.Field	Description	Replication
LRN:: COMMON_ACCESS_CTRL	Controls CSR access to MAC table.	1

Table 97 • **MAC Table Access Registers *(continued)***

Target::Register.Field	Description	Replication
LRN::MAC_ACCESS_CFG_0	Configures MAC (most significant bits) and FID for MAC table entries.	1
LRN::MAC_ACCESS_CFG_1	Configures MAC address (Least significant bits) for MAC table entries and FID.	1
LRN::MAC_ACCESS_CFG_2	Configures the following MAC entry fields: ADDR ADDR_TYPE SRC_KILL_FWD NXT_LRN_ALL CPU_COPY VLAN_IGNORE AFE_FLAG AGE_INTERNAL MIRROR LOCKED VLD	1
LRN::EVENT_STICKY	Sticky status for access performed.	1
LRN::SCAN_NEXT_CFG	MAC table scan filter controls.	1
LRN::SCAN_NEXT_CFG_1	MAC table port move handles when performing MAC table SCAN.	1
ANA_L2::SCAN_FID_CTRL	Controls use of additional FIDs when doing scan.	1
ANA_L2::SCAN_FID_CFG	Configures additional VID/FID filters during scan if enabled via ANA_L2::SCAN_FID_CTRL.	16
LRN::SCAN_LAST_ROW_CFG	Configures a last row applicable for scan.	1
LRN::SCAN_NEXT_CNT	MAC table status when performing MAC table SCAN.	1
LRN::LATEST_POS_STATUS	Holds the latest CPU accessed MAC table location after a CPU_ACCESS_CMD has finished.	1

CPU access to the MAC table is performed by an indirect command-based interface where a number of fields are configured (in LRN::MAC_TABLE_ACCESS_CFG_*) followed by specifying the command (in LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_CMD). The access is initiated by a shot bit that is cleared upon completion of the command (LRN::COMMON_ACCESS_CTRL.MAC_TABLE_ACCESS_SHOT). It is possible to trigger CPU interrupt upon completion (ANA_L2:COMMON:INTR.LRN_ACCESS_COMPLETE_INTR).

The following table lists the types of access possible.

Table 98 • **MAC Table Access Commands**

Command	Purpose	Use
Learn	Insert/learn new entry in MAC table. Position given by HASH(MAC, FID).	Configure MAC and FID of the new entry in LRN::MAC_ACCESS_CFG_0 and LRN::MAC_ACCESS_CFG_1. Configure entry fields in LRN::MAC_ACCESS_CFG_2. Status of operation returned in: LRN::EVENT_STICKY.CPU_LRN_REFRESH_STICKY, LRN::EVENT_STICKY.CPU_LRN_INSERT_STICKY, LRN::EVENT_STICKY.CPU_LRN_REPLACE_STICKY, LRN::EVENT_STICKY.CPU_LRN_REPLACE_FAILED_STICKY, LRN::EVENT_STICKY.CPU_LRN_FAILED_STICKY, and LRN::EVENT_STICKY.LRN_MOVE_STICKY
Unlearn/Forget	Delete/unlearn entry in MAC table given by (MAC, FID). Position given by HASH(MAC, FID).	Configure MAC and FID of the entry to be unlearned in LRN::MAC_ACCESS_CFG_0 and LRN::MAC_ACCESS_CFG_1. Status of operation returned in: LRN::EVENT_STICKY.CPU_UNLEARN_STICKY and LRN::EVENT_STICKY.CPU_UNLEARN_FAILED_STICKY.
Lookup	Lookup entry in MAC table given by (MAC, FID). Position given by HASH(MAC, FID)	Configure MAC and FID of the entry to be looked up in LRN::MAC_ACCESS_CFG_0 and LRN::MAC_ACCESS_CFG_1. Status of lookup operation returned in: LRN::EVENT_STICKY.CPU_LOOKUP_STICKY and LRN::EVENT_STICKY.CPU_LOOKUP_FAILED_STICKY Entry fields are returned in LRN::MAC_ACCESS_CFG_2 if lookup succeeded.
Read direct	Read entry in MAC table indexed by (row, column)	Configure the index of the entry to read in LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_ROW, LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_COL and LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_TYPE Status of read operation returned in LRN::EVENT_STICKY.CPU_READ_DIRECT_STICKY Entry fields are returned in LRN::MAC_ACCESS_CFG_0, LRN::MAC_ACCESS_CFG_1 and LRN::MAC_ACCESS_CFG_2 if read succeeded.

Table 98 • **MAC Table Access Commands *(continued)***

Command	Purpose	Use
Write direct	Write entry in MAC table indexed by (row, column)	Configure the index of the entry to write in LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_ROW, LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_COL and LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_TYPE Configure entry fields in LRN::MAC_ACCESS_CFG_0, LRN::MAC_ACCESS_CFG_1 and LRN::MAC_ACCESS_CFG_2 Status of write operation returned in LRN::EVENT_STICKY.CPU_WRITE_DIRECT_STICKY
Scan	Find next/Find all Searches through the MAC table and either stops at the first row with entries matching the specified filter conditions or perform the specified actions on all entries matching the specified filter conditions.	Configure the starting row for the scan in: LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_ROW, LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_COL and LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_TYPE Configure an ending row for the scan (if searching through the complete table is not required): LRN::SCAN_LAST_ROW_CFG.SCAN_LAST_ROW For information about search filters and corresponding actions, see SCAN Command, page 182.
Find smallest	Get the smallest entry in the MAC table numerically larger than the specified (FID, MAC). FID and MAC are evaluated as a 61 bit number with the FID being most significant.	Configure MAC and FID of the starting point for the search in LRN::MAC_ACCESS_CFG_0 and LRN::MAC_ACCESS_CFG_1. Configure search filters to be used during find smallest: Use LRN::SCAN_NEXT_CFG.SCAN_NEXT_IGNORE_LOCKED_ENA to only search among entries with LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_LOCKED cleared. Use LRN::SCAN_NEXT_CFG.FID_FILTER_ENA to only search among entries with the VID/FID specified in LRN::MAC_ACCESS_CFG_0.MAC_ENTRY_FID. Use LRN::SCAN_NEXT_CFG.ADDR_FILTER_ENA to only search among entries with the address specified in LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_ADDR and LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_ADDR_TYPE
Clear all	Initialize the table	Clears the entire table.

3.20.3.1 Adding a Unicast Entry

Use the following steps to add a unicast entry to the MAC table using the CPU interface.

1. Configure the entry MAC address: LRN::MAC_ACCESS_CFG_0.MAC_ENTRY_MAC_MSB <16 MS MAC bits > LRN::MAC_ACCESS_CFG_1.MAC_ENTRY_MAC_LSB < 32 LS MAC bit >
2. Configure the VLAN FID:
   LRN::MAC_ACCESS_CFG_0.MAC_ENTRY_FID <fid value>
3. Configure the UPSID/UPSPN value:
   LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_ADDR_TYPE 0 (=UPSID)
   LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_ADDR <UPSPN/port value >
4. Make the entry valid: LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_VLD 1
5. Perform access:
   LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_CMD 0 (=LEARN)
   LRN::COMMON_ACCESS_CTRL.MAC_TABLE_ACCESS_SHOT 1
6. Verify that access succeeded: LRN::COMMON_ACCESS_CTRL.MAC_TABLE_ACCESS_SHOT == 0? LRN::EVENT_STICKY.CPU_LRN_INSERT_STICKY==1?

3.20.3.2 Adding Multicast Entry with Destination Set in PGID Table

Use the following steps to add Layer 2 multicast entries to the MAC table using the CPU interface.

1. Configure the entry MAC address: LRN::MAC_ACCESS_CFG_0.MAC_ENTRY_MAC_MSB <16 MSB MAC bits > LRN::MAC_ACCESS_CFG_1.MAC_ENTRY_MAC_LSB < 32 LSB MAC bits >
2. Configure the VLAN ID:
   LRN::MAC_ACCESS_CFG_0.MAC_ENTRY_FID <vid value>
3. Configure the PGID pointer:
   LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_ADDR_TYPE 3 (=MC_IDX)
   LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_ADDR <pgid value >
4. Lock the entry (not subject to aging and automatic removal):
   LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_LOCKED 1
5. Make the entry valid:
   LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_VLD 1
6. Perform access:
   LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_CMD 0 (=LEARN)
   LRN::COMMON_ACCESS_CTRL.MAC_TABLE_ACCESS_SHOT 1
7. Verify that access succeeded: LRN::COMMON_ACCESS_CTRL.MAC_TABLE_ACCESS_SHOT == 0? LRN::EVENT_STICKY.CPU_LRN_INSERT_STICKY==1?

3.20.4 SCAN Command

The SCAN command easily performs table management, such as port flushing, CPU-based aging, and port moves.

Scan can be configured with a variety of filters and modifiers that allows the CPU to easily identify and optionally modify entries matching the configured filter conditions.

The scan operates by running through all entries and whenever entries are found with matching conditions, the scan either stops with a status of the matching location (for example, a “find next” operation) or performs the configured modification actions for all entries (for example, “scan all” operation).

When the scan runs as “find next”, the scan commands are performed for finding the next matching entry. In this case it is possible to specify a starting row for a scan (LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_ROW and LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_TYPE).

Whenever the scan stops with a matching entry, the CPU can process the entry and then sets the starting row to the row following the matching row and, issue another scan command. This can then be repeated until all entries have been scanned.

SCAN is started by setting LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_CMD 5 (=SCAN) and setting the shot bit LRN::COMMON_ACCESS_CTRL.MAC_TABLE_ACCESS_SHOT

SCAN is completed when the shot bit is cleared in LRN::COMMON_ACCESS_CTRL.MAC_TABLE_ACCESS_SHOT.

The following table lists the supported SCAN filters.

Table 99 • **Scan Filters**

Filter Type	Target::Register.Field	Description
FID filter	LRN::SCAN_NEXT_CFG.FID_FILTER_ENA LRN::MAC_ACCESS_CFG_0.MAC_ENTRY_FID ANA_L2::SCAN_FID_CTRL ANA_L2::SCAN_FID_CFG	Finds entries with a specific VID/FID.
Port or Address filter	LRN::SCAN_NEXT_CFG.ADDR_FILTER_ENA, LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_ADDR, LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_ADDR_TYPE LRN::SCAN_NEXT_CFG_1.SCAN_ENTRY_ADDR_MASK	Finds entries with a specific ADDR_TYPE and ADDR. Note that this can be combined with ADDR wildcards; for example, to find all entries related with a given ADDR_TYPE, and so on.
Automatic age scan port filter	LRN::SCAN_NEXT_CFG.SCAN_USE_PORT_FILTER_ENA ANA_L2::FILTER_OTHER_CTRL ANA_L2::FILTER_LOCAL_CTRL	Configures a port filter used for defining which ports are applicable for automatic aging.
Non-locked only filter	LRN::SCAN_NEXT_CFG.SCAN_NEXT_IGNORE_LOCKED_ENA	Finds non locked entries.
Aged only filter	LRN::SCAN_NEXT_CFG.SCAN_NEXT_AGED_ONLY_ENA	Finds only aged entries, that is, AGE_FLAG equal to maximum (configured in ANA_L2::LRN_CFG.AGE_SIZE).
AGE_INTERVAL filter	LRN::SCAN_NEXT_CFG.SCAN_AGE_INTERVAL_MASK	Finds entries with a specific AGE_INTERVAL.
AGE_FLAG filter	LRN::SCAN_NEXT_CFG.SCAN_AGE_FILTER_SEL MAC_ACCESS_CFG_2.MAC_ENTRY_AGE_FLAG	Finds only entries with a specific value.
NXT_LRN_ALL filter	LRN::SCAN_NEXT_CFG.NXT_LRN_ALL_FILTER_ENA MAC_ACCESS_CFG_2. MAC_ENTRY_NXT_LRN_ALL	Finds only entries with a specific value.
Ignore locked filter	LRN::SCAN_NEXT_CFG.SCAN_NEXT_IGNORE_LOCKED_ENA	Finds all or only non locked entries.
Aged only filter	LRN::SCAN_NEXT_CFG.SCAN_NEXT_AGED_ONLY_ENA	Finds all or only aged entries (AGE_FLAG != 0).

The following table lists the available SCAN related actions.

Table 100 • **Scan Modification Actions**

Action Type	Target::Register.Field	Description
Scan next until found	LRN::SCAN_NEXT_CFG.SCAN_NEXT_UNTIL_FOUND_ENA	Controls if scan stops when finding an entry or scans through all entries.
Update AGE_FLAG	LRN::SCAN_NEXT_CFG.SCAN_AGE_FLAG_UPDATE_SEL	Updates AGE_FLAG for found entries. See LRN::SCAN_NEXT_CFG.
Update NXT_LRN_ALL	LRN::SCAN_NEXT_CFG.SCAN_NXT_LRN_ALL_UPDATE_SEL	Update sNXT_LRN_ALL for found entries. See LRN::SCAN_NEXT_CFG.
Change address (MOVE)	LRN::SCAN_NEXT_CFG.SCAN_NEXT_MOVE_FOUND_ENA LRN::SCAN_NEXT_CFG_1.PORT_MOVE_NEW_ADDR LRN::SCAN_NEXT_CFG_1.SCAN_ENTRY_ADDR_MASK	Updates address for found entries.

Table 100 • **Scan Modification Actions *(continued)***

Action Type	Target::Register.Field	Description
Remove	LRN::SCAN_NEXT_CFG.SCAN_NEXT_REMOVE_FOUND_ENA	Removes found entries.
CPU age scan	LRN::SCAN_NEXT_CFG.SCAN_NEXT_INC_AGE_BITS_ENA	Performs CPU based age scan. Aged entries. For example, entries with AGE_FLAG equal to maximum (configured in ANA_L2::LRN_CFG.AGE_SIZE) are removed and Non aged entries, such as entries with AGE_FLAG less than maximum, gets the AGE_FLAG incremented. Automatic Learning, page 187.

The following SCAN status is available:

- Match scan result for the last matching row is reported (LRN::LATEST_POS_STATUS.SCAN_NEXT_STATUS).
- Number of entries found during the scan found (LRN::SCAN_NEXT_CNT.SCAN_NEXT_CNT).
- Scan remove is reported via the sticky event (LRN::EVENT_STICKY.SCAN_REMOVED_STICKY)
- Scan match found is reported via the sticky event (LRN::EVENT_STICKY.ROW_WITH_SCAN_ENTRY_STICKY).

3.20.4.1 Initiating a Port Move for a Specific FID

The following example initiates a port move for specific FID.

- Move found entries:
  Set LRN::SCAN_NEXT_CFG.SCAN_NEXT_MOVE_FOUND_ENA = 1.
- Specify the FID filter:
  Set LRN::SCAN_NEXT_CFG.FID_FILTER_ENA = 1.
  Set LRN::MAC_ACCESS_CFG_0.MAC_ENTRY_FID = <FID value>.
- Specify port filter:
  Set LRN::SCAN_NEXT_CFG.ADDR_FILTER_ENA = 1.
  Set LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_ADDR_TYPE = 0 to specify address type UPSID_PN.
  Set LRN::MAC_ACCESS_CFG_2.MAC_ENTRY_ADDR = <UPSID, UPSPN> to specify UPSID and UPSPN.
- Specify the new address:
  Set LRN::SCAN_NEXT_CFG_1.PORT_MOVE_NEW_ADDR = <new UPSID, new UPSPN>
  Set LRN::SCAN_NEXT_CFG_1.SCAN_ENTRY_ADDR_MASK = all ones.
- Scan through all rows by starting at row 0:
  Set LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_ROW = 0.
  Set LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_DIRECT_TYPE = 0.
- Issue SCAN command:
  Set LRN::COMMON_ACCESS_CTRL.CPU_ACCESS_CMD = 5.
  Set LRN::COMMON_ACCESS_CTRL.MAC_TABLE_ACCESS_SHOT = 1.
- Wait until LRN::COMMON_ACCESS_CTRL.MAC_TABLE_ACCESS_SHOT is cleared.

3.20.5 Forwarding Lookups

This section describes forwarding handling. The following table lists the applicable registers.

Table 101 • **Forwarding Registers**

Target::Register.Field	Description	Replication
ANA_L2:COMMON:FWD_CFG	Configures common forwarding options	1

3.20.5.1 DMAC-Based Forwarding

The analyzer performs destination lookup in the MAC table to determine if the destination MAC address is known or unknown.

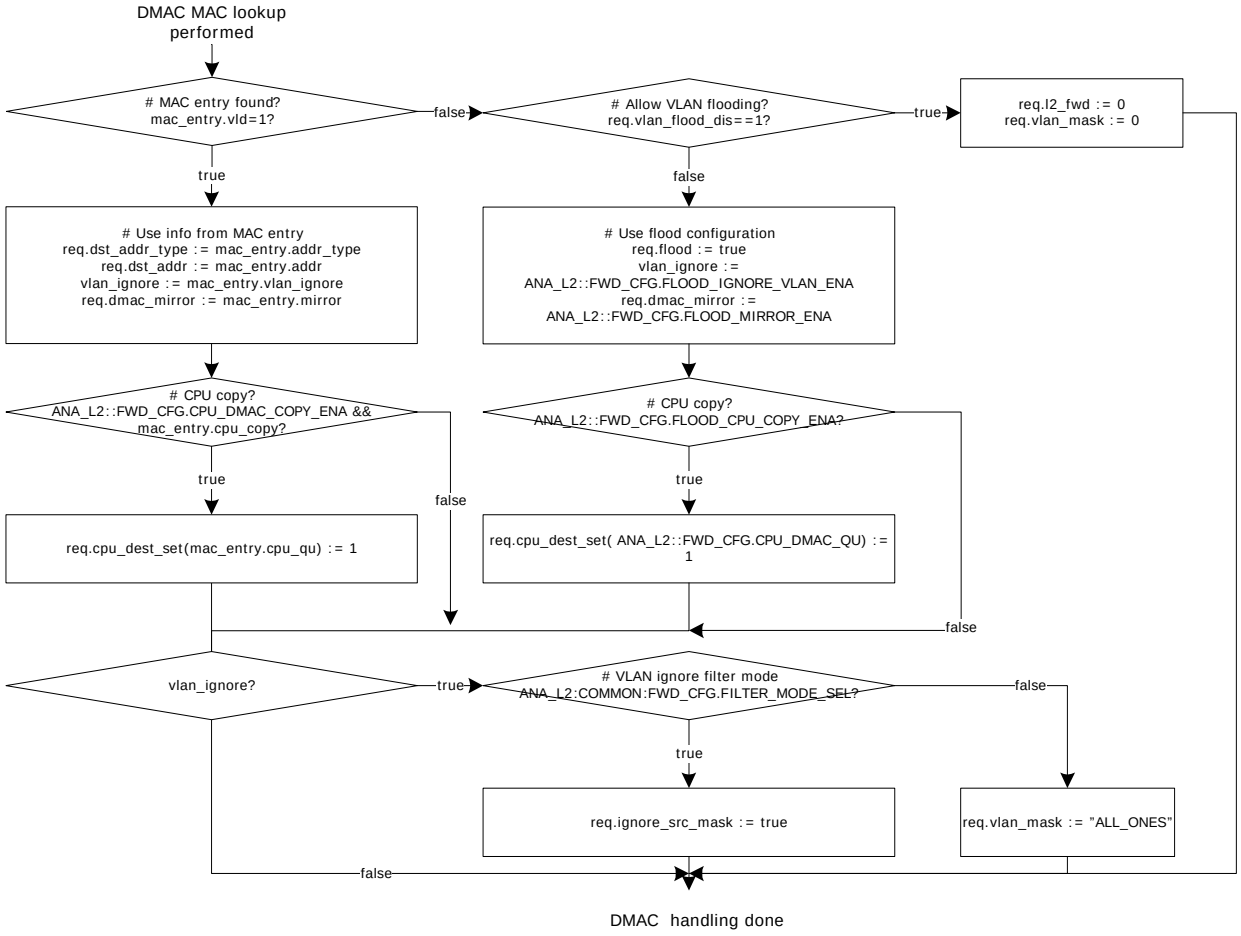
- If the destination is unknown, a flood forwarding decision is made if forwarding from unknown sources is allowed (ANA_L3:VLAN:VLAN_CFG.VLAN_SEC_FWD_ENA). The associated flood masks are located in the PGID table.
- If the destination address is known, the associated ADDR_TYPE and ADDR stored in the destination entry is used to find the destination port or ports in the PGID table.

Flooded traffic can be made available for mirroring (ANA_L2:COMMON:FWD_CFG.FLOOD_MIRROR_ENA) and CPU copying (ANA_L2:COMMON:FWD_CFG.FLOOD_CPU_COPY_ENA ANA_L2:COMMON:FWD_CFG.CPU_DMAC_QU).

Traffic with known DMAC and VLAN_IGNORE set LRN:COMMON:MAC_ACCESS_CFG_2.MAC_ENTRY_VLAN_IGNORE for CPU learned frames or LRN:COMMON:AUTO_LRN_CFG.AUTO_LRN_IGNORE_VLAN for auto learned entries) and optionally also flooded traffic (when ANA_L2:COMMON:FWD_CFG.FLOOD_IGNORE_VLAN_ENA set) can be configured to ignore VLAN port mask or source port mask (ANA_L2:COMMON:FWD_CFG.FILTER_MODE_SEL).

The DMAC lookup of the received {DMAC, EFID} in the MAC table is shown in the following illustration.

Figure 47 • **DMAC Lookup**



3.20.6 Source Check and Automated Learning

This section describes hardware (or automated) learning related to source checking. The following table lists the applicable registers.

Table 102 • **Hardware-Based Learning**

Target::Register.Field	Description	Replication
ANA_L2::AUTO_LRN_CFG	Configures automatic learning per port	1
ANA_L2::LRN_SECUR_CFG	Configures secure forwarding per port	1
ANA_L2::LRN_SECUR_LOCKED_CFG	Configures secure forwarding for static locked entries per port.	1
ANA_L2::LRN_COPY_CFG	Configures CPU copy of learn frames per port.	1
ANA_L2::MOVELOG_STICKY	Identifies ports with moved stations.	1
ANA_L2::LRN_CFG	Configures common learning properties.	1
ANA_L3:VLAN:VLAN_CFG.VLAN_FID	Configures filtering identifier (FID) to be used for learning.	Per VLAN
LRN::AUTO_LRN_CFG	Configures automatic learning options.	1
LRN::EVENT_STICKY	Signal various sticky learning events.	1
ANA_L2:PORT_LIMIT:PORT_LIMIT_CTRL	Controls automatic learn limits per logical port and GLAG.	per port and GLAG (11+8 instances)
ANA_L2:PORT_LIMIT:PORT_LIMIT_STATUS	Status for port limits.	per port and GLAG (11+8 instances)
ANA_L2:LRN_LIMIT:FID_LIMIT_CTRL	Controls automatic learn limits per FID.	per FID (4,224 instances)
ANA_L2:LRN_LIMIT:FID_LIMIT_STATUS	Status for FID limits.	per FID (4,224 instances)
ANA_L2:ISDX:SERVICE_CTRL.ISDX_BASED_SRC_ENA	Enable service-based learning.	Per ISDX

The device learns addresses from each received frame's SMAC field, so that subsequently forwarded frames whose destination addresses have been learned can be used to restrict transmission to the port or ports necessary to reach their destination.

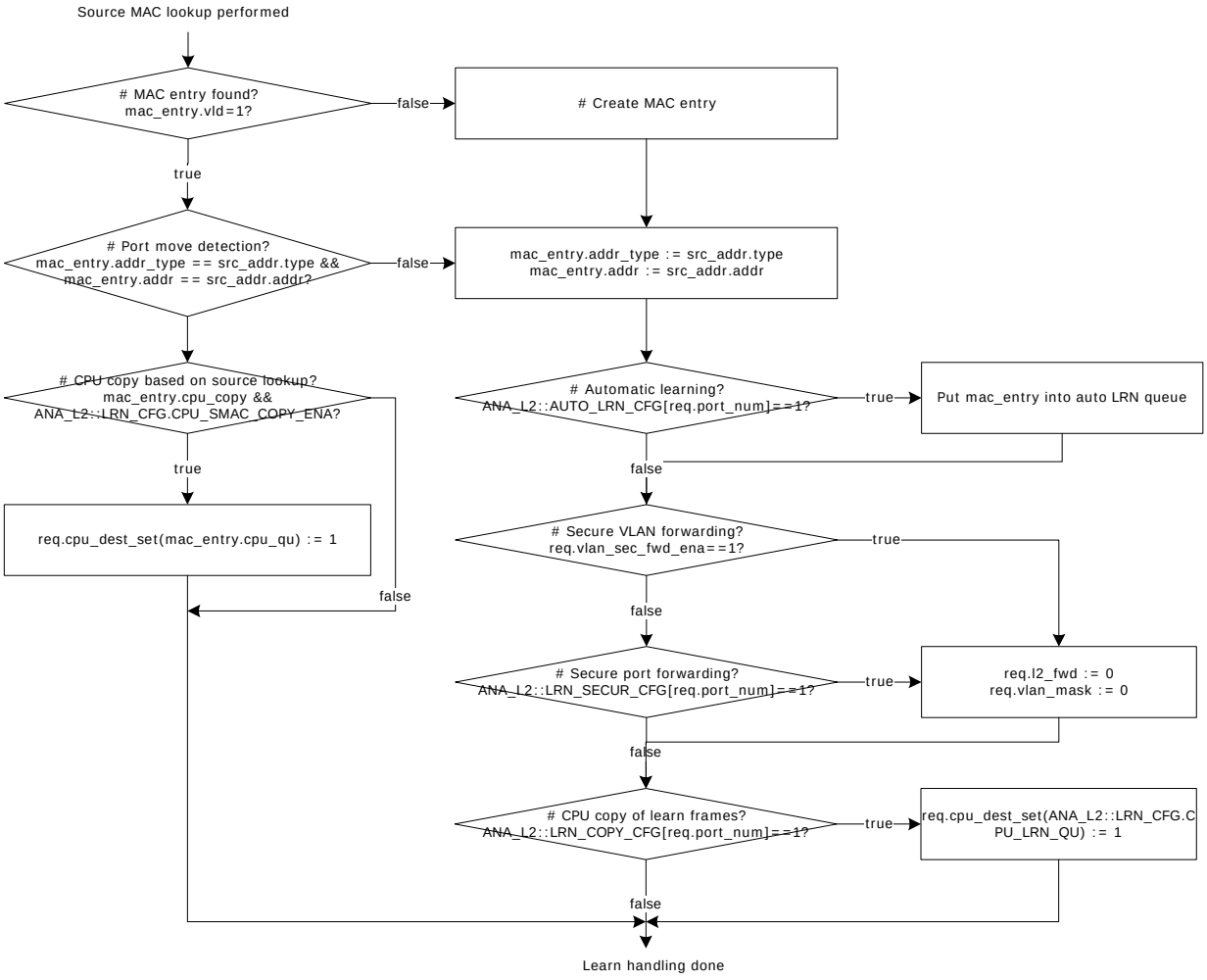
The analyzer performs source lookup of the received {SMAC, IFID} in the MAC table to determine if the source station is known or unknown. For more information, see Figure 48, page 187. If the source station is known, a port move detection is performed by checking that the frame was received at the expected port as specified by the ADDR_TYPE and ADDR stored in the source entry. The expected port can be a logical port (represented as UPSID, UPSPN), an MC_IDX, or a global aggregated port (represented as GLAGID). It is possible to control the received address per services (ANA_L2:ISDX:SERVICE_CTRL.ISDX_BASED_SRC_ENA, ANA_L2:ISDX:SERVICE_CTRL.FWD_TYPE and ANA_L2:ISDX:SERVICE_CTRL.FWD_ADDR). Failing source checks can trigger a number of associated actions:

- Secure forwarding. Disable forwarding from unknown or moved stations. This action can be triggered by two configurations:
  - VLAN-based Secure forwarding controlled per VLAN (ANA_L3:VLAN:VLAN_CFG.VLAN_SEC_FWD_ENA).
  - Port-based secure forwarding controlled per port (ANA_L2::LRN_SECUR_CFG).
- Secure forwarding from locked entries. Disable forwarding from known but moved stations with locked MAC table entry (ANA_L2:COMMON:LRN_SECUR_LOCKED_CFG).

- Automatic learning enabled per port (ANA_L2::AUTO_LRN_CFG). Automatic learning of traffic received from an unknown source station will cause the MAC table to be updated with a new entry. Traffic received from a known source station with changed port will cause the known MAC table entry ADDR_TYPE and ADDR to be updated.
- CPU copy enabled per port (ANA_L2::LRN_COPY_CFG and ANA_L2::LRN_CFG.CPU_LRN_QU). A learn copy is sent to CPU.

These actions are controlled independently. In other words, it is possible to send a CPU copy as well as disable normal forwarding.

Figure 48 • **Source Check**



The source lookup is also used to clear an Age flag (used as activity indication) in the MAC table to allow inactive MAC table entries to be pruned.

Port move detection is not performed for locked MAC table entries unless enabled (ANA_L2::LRN_CFG.LOCKED_PORTMOVE_DETECT_ENA).

Frames failing locked port move detection can be copied to CPU queue (ANA_L2::LRN_CFG.CPU_LRN_QU and ANA_L2::LRN_CFG.LOCKED_PORTMOVE_COPY_ENA).

3.20.6.1 Automatic Learning

The device can automatically add or update entries in the MAC table. New entries added by automatic learning are formatted as follows.

- VLD is set
- MAC is set to the frame's SMAC address

- FID set to the frame's REQ.FID
- ADDR_TYPE, ADDR is set to:
  UPSID_PN if received on a non aggregated port
  GLAG if received on a global aggregated port
  ANA_L2:ISDX:SERVICE_CTRL.FWD_TYPE and ANA_L2:ISDX:SERVICE_CTRL.FWD_ADDR if service based source is enabled (ANA_L2:ISDX:SERVICE_CTRL.ISDX_BASED_SRC_ENA)
- MAC_CPU_COPY (LRN::AUTO_LRN_CFG.AUTO_LRN_CPU_COPY)
- MAC_CPU_QU (LRN::AUTO_LRN_CFG.AUTO_LRN_CPU_QU)
- SRC_KILL (LRN::AUTO_LRN_CFG.AUTO_LRN_SRC_KILL_FWD)
- IGNORE_VLAN (LRN::AUTO_LRN_CFG.AUTO_LRN_IGNORE_VLAN)
- MIRROR (LRN::AUTO_LRN_CFG.AUTO_LRN_MIRROR)
- AGE_INTERVAL (LRN::AUTO_LRN_CFG.AUTO_AGE_INTERVAL)
- All other fields are cleared

When a frame is received from a known station, that is, the MAC table already contains an entry for the received frame's (SMAC, VID/FID), the analyzer clears the AGED_FLAG for unlocked entries (LOCKED flag cleared) and optionally, locked entries (ANA_L2:COMMON:LRN_CFG.AGE_LOCKED_ENA). A cleared AGE_FLAG implies that the station is active. For more information, Automated Aging (AUTOAGE), page 189.

For unlocked entries, ADDR_TYPE and ADDR fields are compared against the following:

- UPSID_PN if received on a non aggregated port
- GLAG if received on a global aggregated port
- ANA_L2:ISDX:SERVICE_CTRL.FWD_TYPE and ANA_L2:ISDX:SERVICE_CTRL.FWD_ADDR if service based source is enabled (ANA_L2:ISDX:SERVICE_CTRL.ISDX_BASED_SRC_ENA)

If there is a difference, the source entry ADDR_TYPE and ADDR is updated, which implies the station has moved to a new port.

Source check for entries stored with ADDR_TYPE=MC_IDX (=3) can be ignored (ANA_L2::LRN_CFG.IGNORE_MCIDX_PORTMOVE_ENA).

3.20.6.2 Port Move Log

A port move occurs when an existing MAC table entry for (MAC, FID) is updated with new port information (ADDR_TYPE and ADDR). Such an event is logged and reported (in ANA_L2::MOVELOG_STICKY) by setting the bit corresponding to the new port.

If port moves continuously occurs, it may indicate a loop in the network or a faulty link aggregation configuration.

Port moves for locked entries can be detected in ANA_L2::LRN_CFG.LOCKED_PORTMOVE_DETECT_ENA.

3.20.6.3 Port Learn Limits

It is possible to specify the maximum number of entries in the MAC table per port.

Port limits can be specified per logical port (local link aggregated port) and per global link aggregated port (GLAG) (ANA_L2:PORT_LIMIT:PORT_LIMIT_CTRL.PORT_LRN_CNT_LIMIT). These limits do not take multicast into account.

The current number of learned entries are always available in ANA_L2:PORT_LIMIT:PORT_LIMIT_STATUS.PORT_LRN_CNT and are updated whenever either an automatic learn operation or a CPU access is performed.

After a limit is reached, both automatic learning and CPU-based learning of unlocked entries are denied. A learn frame to be learned when the limit is reached can selectively be discarded, redirected to CPU, or copied to CPU (controlled in ANA_L2:PORT_LIMIT:PORT_LIMIT_CTRL.PORT_LIMIT_EXCEED_SEL). Trying to exceed the configured limit can also trigger interrupt (ANA_L2:PORT_LIMIT:PORT_LIMIT_CTRL.PORT_LIMIT_EXCEED_IRQ_ENA).

When learn limits are exceeded, a corresponding sticky bit is set (ANA_L2:PORT_LIMIT:PORT_LIMIT_STATUS.PORT_LRN_LIMIT_EXCEEDED_STICKY).

If a MAC table entry interrupt moves from one port to another port, it is also reflected in the current number of learned entries.

3.20.6.4 Filtering Identifier Learn Limits

It is possible to specify the maximum number of entries to be used per FID (ANA_L2:LRN_LIMIT:FID_LIMIT_CTRL.FID_LRN_CNT_LIMIT). These limits do not take multicast into account.

The current number of learned entries are always available in ANA_L2:LRN_LIMIT:FID_LIMIT_STATUS.FID_LRN_CNT.

After a limit is reached, both automatic learning and CPU-based learning of unlocked entries are denied. A learn frame to be learned when the limit is reached can selectively be discarded, redirected to CPU, or copied to CPU (controlled by ANA_L2:LRN_LIMIT:FID_LIMIT_CTRL.FID_LIMIT_EXCEED_SEL). Attempts to exceed the configured limit can also trigger interrupt (ANA_L2:LRN_LIMIT:FID_LIMIT_CTRL.FID_LIMIT_EXCEED_IRQ_ENA).

When the learn limits are exceeded, a corresponding sticky bit is set (ANA_L2:LRN_LIMIT:FID_LIMIT_STATUS.FID_LRN_LIMIT_EXCEEDED_STICKY).

3.20.6.5 Shared or Independent VLAN Learning in MAC Table

The device can be set up to do a mix of Independent VLAN Learning (IVL), where MAC addresses are learned separately on each VLAN and Shared VLAN Learning (SVL), where a MAC table entry is shared among a group of VLANs. For shared VLAN learning, a MAC address and a filtering identifier (FID) define each MAC table entry. A set of VIDs used for Shared VLAN learning maps to the same FID. Shared VLAN learning is only applicable for unicast traffic.

Addresses learned from frames with one VLAN Identifier (VID) may or may not be used to filter frames with the same SMAC but another VID, depending on management controls. If learned information is shared for two or more given VIDs those VIDs map to a single Filtering Identifier (FID) and the term Shared VLAN Learning (SVL) is used to describe their relationship. If the information is not shared, the VIDs map to different FIDs and Independent VLAN Learning (IVL) is being used. FID is configured per VLAN (ANA_L3:VLAN:VLAN_CFG.VLAN_FID).

For example, if VID 16 and VID 17 use SVL and share FID 16, then a MAC entry (MAC #X, FID 16) is known for both VID 16 and VID 17.

If VID 16 and VID 17 use IVL and use FID 16 and FID 17, a MAC entry (MAC #X, FID 16) is only known for VID 16; that is, the MAC address will be unknown for VID 17 (FID = 17).

In a VLAN-unaware operation, the device is set up to do SVL; that is, MAC addresses are learned for all VLANs. Protocol-based VLAN, where traffic is classified into multiple VLANs based on protocol, requires SVL

3.20.7 Automated Aging (AUTOAGE)

This section describes how to configure automated age scanning. The following table lists the applicable registers.

Table 103 • **Automated Age Scan Registers Overview**

Target::Register.Field	Description	Replication
LRN::AUTOAGE_CFG	Configures automated age scan of MAC table for each of the four AGE_INTERVALs.	4
LRN::AUTOAGE_CFG_1	Configures automated age scan of MAC table.	1
LRN::AUTOAGE_CFG_2	Configures automated age scan of MAC table.	1
LRN::EVENT_STICKY. AUTOAGE_SCAN_COMPLETED_STICKY	Signals completion of an automatic scan.	1
LRN::EVENT_STICKY. AUTOAGE_SCAN_STARTED_STICKY	Signals start of an automatic scan.	1

Table 103 • **Automated Age Scan Registers Overview *(continued)***

Target::Register.Field	Description	Replication
LRN::EVENT_STICKY. AUTOAGE_AGED_STICKY	Signals entries aged due to automatic aging.	1
LRN::EVENT_STICKY. AUTOAGE_REMOVE_STICKY	Signals entries removed due to automatic aging.	1
ANA_L2::FILTER_LOCAL_CTRL	Configures front port filters to be used by automatic aging and CPU scan.	1
ANA_L2::FILTER_OTHER_CTRL	Configures remote scan filter to be used by automatic aging and CPU scan.	1
ANA_L2::LRN_CFG.AGE_SIZE	Configures the AGE_FLAG size.	1

Entry AGE_FLAG is used to detect inactive sources. The AGE_FLAG field is cleared upon receiving traffic from a given source station and is periodically incremented by hardware to detect and delete inactive source entries.

The automated age scan scans the MAC table and checks age candidates (entries with LOCK flag set to 0) for activity.

If an entry's AGE_FLAG is set to maximum value (configured in ANA_L2::LRN_CFG.AGE_SIZE), the entry is deemed inactive and removed. If an entry's AGE_FLAG is less than the maximum value, the AGE_FLAG is incremented.

The flag is cleared when receiving frames from the station identified by the MAC table entry.

It is possible to configure and forget AUTOAGE, which means that once configured automated aging is performed without any further CPU assistance.

The interval between automatic aging can be controlled by specifying AGE_FLAG size (ANA_L2::LRN_CFG.AGE_SIZE), unit size (LRN::AUTOAGE_CFG.UNIT_SIZE), and the number of units between aging (LRN::AUTOAGE_CFG.PERIOD_VAL).

Setting LRN::AUTOAGE_CFG.UNIT_SIZE different from 0 to start the automatic aging.

To temporarily disable automatic aging, set LRN::AUTOAGE_CFG. PAUSE_AUTO_AGE_ENA to 1. Additional control of automatic aging allows for instantaneously start and stop of current age scan (LRN::AUTOAGE_CFG_1.FORCE_HW_SCAN_SHOT and LRN::AUTOAGE_CFG_1.FORCE_HW_SCAN_STOP_SHOT).

It is also possible to do a controlled stopping of current age scan after it completes (LRN::AUTOAGE_CFG_1.FORCE_IDLE_ENA).

It is possible to selectively do age filtering of local ports specified in ANA_L2::FILTER_LOCAL_CTRL.FILTER_FRONTPORT_ENA or selectively do age filtering of entries learned on remote devices in a multichip configuration (ANA_L2::FILTER_OTHER_CTRL.FILTER_REMOTE_ENA). Both types must be enabled (LRN::AUTOAGE_CFG_1.USE_PORT_FILTER_ENA).

The state of automatic aging can be monitored (LRN::AUTOAGE_CFG_2.NEXT_ROW, LRN::AUTOAGE_CFG_2.SCAN_ONGOING_STATUS, LRN::EVENT_STICKY.AUTOAGE_SCAN_COMPLETED_STICKY and LRN::EVENT_STICKY.AUTOAGE_SCAN_STARTED_STICKY).

It is possible to observe whether entries are affected by a sticky event (LRN::EVENT_STICKY.AUTOAGE_AGED_STICKY and LRN::EVENT_STICKY.AUTOAGE_REMOVE_STICKY).

Example

For a 300-second age time, set four age periods of 75 seconds each:

1. Set ANA_L2::LRN_CFG.AGE_SIZE to 1.
2. Set LRN::AUTOAGE_CFG[0]. AUTOAGE_UNIT_SIZE to 3.

3. Set LRN::AUTOAGE_CFG[0]. AUTOAGE_PERIOD_VAL to 75.

3.20.8 Service Handling

This section describes how to configure services in analyzer Layer 2. The following table lists the applicable registers. Using the ISDX table requires enabling ISDX table lookup in ANA_L2::FWD_CFG.ISDX_LOOKUP_ENA.

Table 104 • **ISDX Registers Overview**

Target::Register.Field	Description	Replication
ANA_L2:COMMON:FWD_CFG.ISDX_LOOKUP_ENA	Enables lookup of ISDX in ISDX table.	1
ANA_L2:COMMON:FWD_CFG.PORT_DEFAULT_BDLB_ENA	Configures use of port-based dual leaky bucket index into ANA_AC_POL:BDLB (ANA_L2::PORT_DLB_CFG.PORT_DLB_IDX) when no service is selected (ISDX=0).	1
ANA_L2:COMMON:FWD_CFG.QUEUE_DEFAULT_SDLB_ENA	Configures use of queue-based dual leaky bucket index into ANA_AC_POL:SDLB (in ANA_L2::PORT_DLB_CFG.QUEUE_DLB_IDX) when no service is selected (ISDX=0).	1
ANA_L2:COMMON:PORT_DLB_CFG.PORT_IDX	Configures per port BDLB indexes.	Per port
ANA_L2::PORT_DLB_CFG.QUEUE_DLB_IDX	Configures per queue SDLB indexes = ANA_L2::PORT_DLB_CFG[port].QUEUE_DLB_IDX + IPRIO.	Per port
ANA_L2:ISDX:SERVICE_CTRL	Configures service forwarding.	Per ISDX
ANA_L2:ISDX:PORT_MASK_CFG	Configures service based port mask.	Per ISDX
ANA_L2:ISDX:MISC_CFG	Controls various indexes such as BDLB and BUM.	Per ISDX
ANA_L2:ISDX:DLB_CFG	Configures service DLB policer.	Per ISDX
ANA_L2:ISDX:DLB_COS_CFG	Configures Service DLB policer offset per COSID.	Per COSID per ISDX
ANA_L2:ISDX:QGRP_CFG	Configures QGRP handling QSYS.	Per ISDX
ANA_L2:ISDX:ISDX_BASE_CFG	Configures service counter index.	Per ISDX
ANA_L2:ISDX:ISDX_COS_CFG	Configures service counter index offset per COSID.	Per COSID per ISDX

3.20.8.1 Service-Based Forwarding

A service port mask can reduce or replace VLAN port mask to limit forwarding to certain ports (ANA_L2:ISDX:SERVICE_CTRL.PORT_MASK_REPLACE_ENA and ANA_L2:ISDX:PORT_MASK_CFG).

Source port filtering can be disabled per service (ANA_L2:ISDX:SERVICE_CTRL.SRC_MASK_DIS).

Normal link aggregation selection can be controlled per service by either disabling or forcing a specific link aggregation contribution (ANA_L2:ISDX:SERVICE_CTRL.AGGR_REPLACE_ENA and ANA_L2:ISDX:SERVICE_CTRL.AGGR_VAL).

It is possible to do service-based forwarding and overrule DMAC-based forwarding (by setting ANA_L2:ISDX:SERVICE_CTRL.ISDX_BASED_FWD_ENA == 1 and ANA_L2:ISDX:SERVICE_CTRL.CDA_FWD_ENA == 0).

Service-based forwarding can either be a flood decision (setting ANA_L2:ISDX:SERVICE_CTRL.FWD_TYPE == 7: NO_ADDR) or a PGID forwarding decision (ANA_L2:ISDX:SERVICE_CTRL.FWD_TYPE == 3 : MC_IDX and

ANA_L2:ISDX:SERVICE_CTRL.FWD_ADDR set to the used mc_idx into the PGID table). PGID forwarding allows control of forwarding for multiple services via a single PGID entry.

Note: ANA_L2:ISDX:SERVICE_CTRL.ISDX_BASED_FWD_ENA cannot be used together with ANA_L2:ISDX:SERVICE_CTRL.ISDX_BASED_SRC_ENA.

3.20.8.2 Service-Based Source Interface

Service-based learning (enabled in ANA_L2:ISDX:SERVICE_CTRL.ISDX_BASED_SRC_ENA) makes it possible to configure which interface a given ISDX is associated with (configured in ANA_L2:ISDX:SERVICE_CTRL.FWD_TYPE and ANA_L2:ISDX:SERVICE_CTRL.FWD_ADDR).

This makes it is possible to learn MAC addresses associated with an interface different from received logical port interface (configured in ANA_CL:PORT:PORT_ID_CFG) for a service.

As an example, MAC addresses can be associated with a MC_IDX for working and protected ISDX in a protected service. This eliminates the need for updating the MAC table when performing linear protection switchover.

3.20.8.3 Ingress Service Bandwidth Profiles

Service bandwidth profile is configured in ANA_AC_POL:SDLB with the index configured in the ISDX table.

Service policing can be configured per ISDX, per COSID allowing either bandwidth profiles per ISDX or per COSID, per ISDX. This is configured as a base pointer index with configurable offset per COSID (ANA_L2:ISDX:DLB_CFG and ANA_L2:ISDX:DLB_COS_CFG).

A queue-based bandwidth profile can be configured for traffic not assigned to a service when ISDX = 0 (ANA_L2::FWD_CFG.QUEUE_DEFAULT_SDLB_ENA and ANA_L2::PORT_DLB_CFG.QUEUE_DLB_IDX).

It is possible to configure a pipeline point (ANA_L2:ISDX:MISC_CFG.PIPELINE_PT) that controls where in the pipeline traffic is policed. For example, traffic discarded or extracted at a pipeline point before the configured pipeline point does not cause service DLB policing and ISDX counter updates.

3.20.8.4 Ingress Bundle Bandwidth Profiles

Bundle bandwidth profile is configured in ANA_AC_POL:BDLB with the index configured per ISDX in ANA_L2:ISDX:MISC_CFG.BDLB_IDX.

A port based bandwidth profile can be configured for traffic not assigned to a service when ISDX = 0 (ANA_L2::FWD_CFG.PORT_DEFAULT_BDLB_ENA and ANA_L2::PORT_DLB_CFG.PORT_DLB_IDX).

3.20.8.5 Broadcast, Unknown and Multicast (BUM) Bandwidth Profiles

BUM bandwidth profile is configured in ANA_AC_POL:BUM_SLB with the index configured per VLAN in ANA_L3:VLAN:BUM_CFG. This index can be overruled per ISDX (through ANA_L2:ISDX:MISC_CFG.BUM_SLB_ENA and ANA_L2:ISDX:MISC_CFG.BUM_SLB_IDX).

3.20.8.6 Ingress Service Statistics

Service statistics in ANA_AC and in QSYS are based on an index configured in the ISDX table.

The ingress service statistics index can be configured per ISDX, per COSID allowing either statistics per ISDX or per COSID per ISDX. This is configured as a base index with configurable offset per COSID (ANA_L2:ISDX:ISDX_BASE_CFG and ANA_L2:ISDX:ISDX_COS_CFG).

Note: Service statistics depend on whether traffic is discarded before or after the pipeline point configured for ISDX (ANA_L2:ISDX:MISC_CFG.PIPELINE_PT).

3.20.9 Interrupt Handling

The following table lists the interrupt handling registers.

Table 105 • **Analyzer Interrupt Handling Registers Overview**

Target::Register.Field	Description	Replication
ANA_L2::INTR	Status of interrupt source	1
ANA_L2::INTR_ENA	Mask interrupt	1
ANA_L2::INTR_IDENT	Status of interrupt	1

A sticky status of all interrupt sources in the analyzer is available (ANA_L2::INTR.VCAP_S2_INTR), and all interrupt source can individually be configured to trigger CPU interrupt (ANA_L2::INTR_ENA). The current CPU interrupt status is available (ANA_L2::INTR_IDENT).

The following analyzer interrupt sources exist:

VCAP IS2 can be setup to trigger CPU interrupt (VCAP_S2_INTR) when an entry with enabled interrupt is hit (VCAP IS2 action INTR_ENA).

It is possible to trigger interrupt (PORT_LRN_LIMIT_INTR) when learning on a port attempt to exceed a configured port learn limit (configured in ANA_L2:PORT_LIMIT:PORT_LIMIT_CTRL.PORT_LIMIT_EXCEED_IRQ_ENA).

It is possible to trigger interrupt (FID_LRN_LIMIT_INTR) when learning on a FID attempt to exceed a configured limit (configured in ANA_L2:LRN_LIMIT:FID_LIMIT_CTRL.FID_LIMIT_EXCEED_IRQ_ENA).

LRN access to the MAC table can be configured to trigger interrupt (LRN_ACCESS_COMPLETE_INTR) after completion. This is useful when multiple CPU scans must be performed as fast as possible, for example, when sorting the MAC addresses extracted from the entire MAC table using SCAN commands. For more information about using the SCAN command, see SCAN Command, page 182.

3.21 Analyzer Access Control Forwarding, Policing, and Statistics

This section provides information about analyzer access control (ANA_AC) forwarding, policing, and statistics.

3.21.1 Mask Handling

This section describes how a number of port masks are applied to ensure correct forwarding.

- VLAN port mask from ANA_L3:VLAN. This mask is used to ensure frames are only forwarded within a dedicated VLAN (or VSI if service forwarded). The VLAN mask handle protection by means of hardware assisted updates. For more information, see VLAN Table Update Engine, page 135. The VLAN mask can be modified by VCAP CLM full action MASK_MODE and by VCAP IS2 action MASK_MODE.
- ISDX port mask from ANA_L2:ISDX:PORT_MASK_CFG. This mask is used for service forwarding and can optionally (ANA_L2:ISDX:SERVICE_CTRL.PORT_MASK_REPLACE_ENA) overrule the use of the VLAN port mask.
- PGID port mask from ANA_AC:PGID. This mask is based on the DMAC lookup in the PGID table. One of the six flood PGID entries is used if the DMAC is unknown. For DMAC known in the MAC table, the associated address is used to select the used PGID entry.
- Source port mask from ANA_AC:SRC. This mask is used to ensure frames are never forwarded to the port it was received on and the used entry is found by looking up the source port.
- Global source port mask from ANA_AC:SRC. This mask is used to ensure frames are never forwarded to the global port it was received on and the used entry is found by looking up the global stacking source port.
- Aggregation port mask from ANA_AC:AGGR. This mask is used to ensure frames are forwarded to one port within each aggregate destination port.

- REQ.port_mask and REQ.mask_mode. This mask is special in the sense that it is a general purpose mask that can be used for various security features and protection. The mask is controlled by VCAP CLM and VCAP IS2.

The following sections provide more information about the different port masks.

3.21.1.1 PGID Lookup

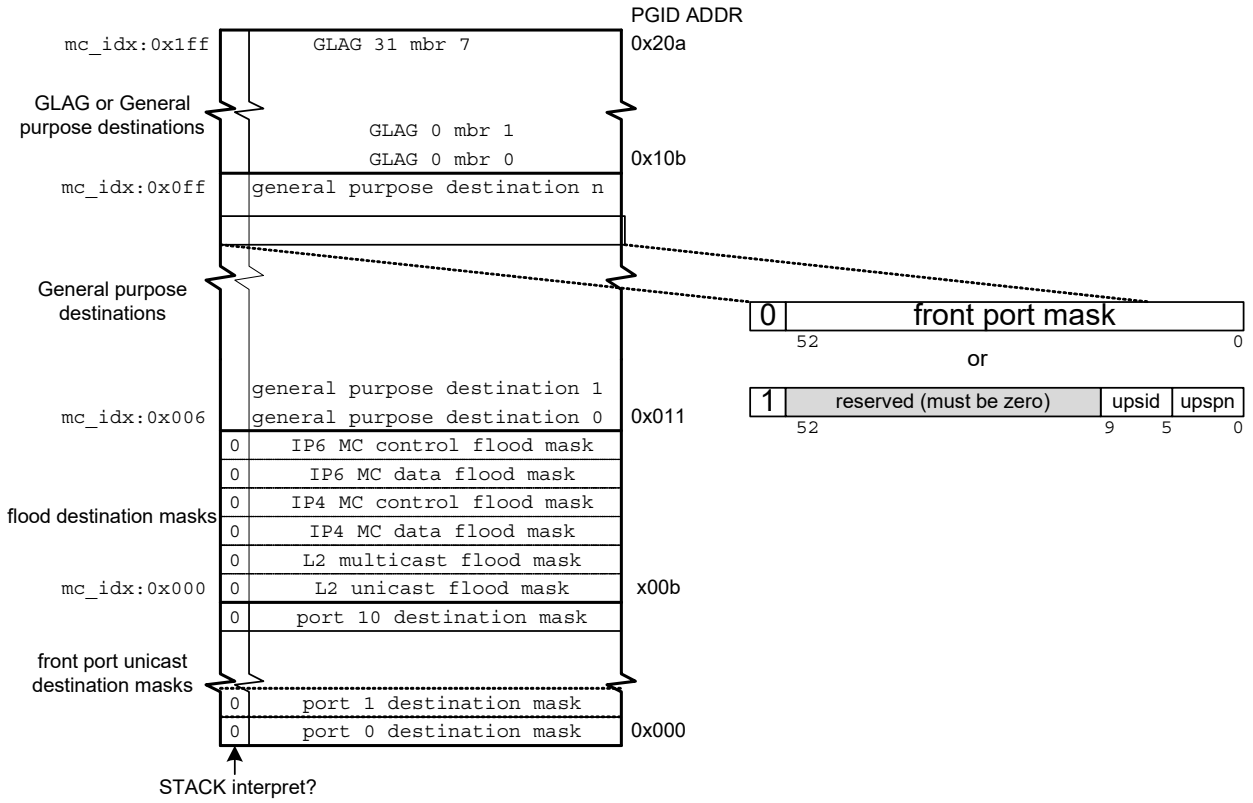
The following table lists the registers associated with PGID lookup.

Table 106 • **PGID Registers Overview**

Target::Register.Field	Description	Replication
ANA_AC:PGID:PGID_MISC_CFG	Configures how to interpret PGID_CFG and configures CPU copy with associated CPU queue.	per PGID
ANA_AC:PGID:PGID_CFG	Configures destination port mask or destination {UPSID,PN}.	per PGID

The PGID table is organized as shown in the following illustration.

Figure 49 • **PGID Layout**



The forwarding process generates a forwarding decision that is used to perform a lookup in the PGID table. The following forwarding decisions are possible.

- Flood forwarding to one of the six flood masks located from address 11 to address 16 in the PGID table is used as destination mask.
- Unicast forwarding to {UPSID,UPSPN}. UPSID is checked against ANA_AC::COMMON_VSTAX_CFG.OWN_UPSID:

  If identical, destination mask is found based on lookup of UPSPN within the front port destinations of the PGID table (address 0 to 10).
  If not identical, destination mask is found based on lookup of UPSID in ANA_AC:UPSID:UPSID_CFG.

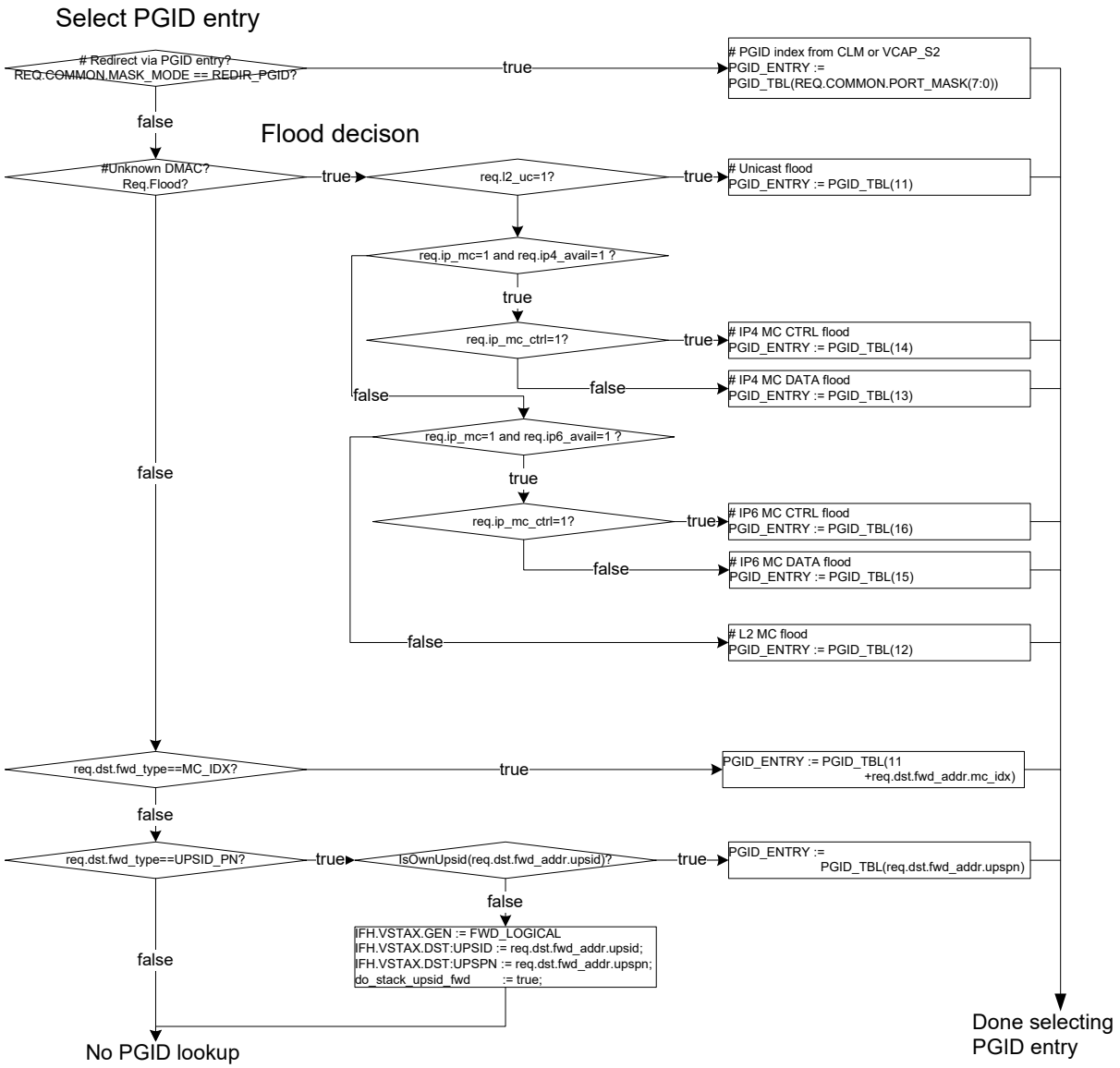
- Multicast forwarding: mc_idx is looked up in the PGID table with an offset of 11. Based on the returned entry stack bit:

  If cleared, the returned entry mask is interpreted as a destination mask.
  If set, the returned entry mask is {UPSID,UPSPN} and the destination mask is found based on lookup of UPSID in ANA_AC:UPSID:UPSID_CFG.
- REQ destination set (REQ.COMMON.PORT_MASK when REQ.COMMON.MASK_MODE=REPLACE_PGID). Destination set (REQ.COMMON.PORT_MASK from VCAP CLM or VCAP IS2 is directly used as destination mask without PGID lookup.
- REQ.L2_FWD cleared. Frame is discarded.

It is possible to generate a CPU copy can be generated when using the PGID lookup. If ANA_AC:PGID:PGID_MISC_CFG.PGID_CPU_COPY_ENA is set in the used PGID entry, a copy is sent to CPU queue specified by ANA_AC:PGID:PGID_MISC_CFG.PGID_CPU_QU.

The following illustration depicts the PGID lookup decision.

Figure 50 • **PGID Lookup Decision Forwarding**



The following debug events can be used to observe current forwarding:

ANA_AC:PS_STICKY:STICKY.PGID_CPU_MASK_STICKY,
ANA_AC:PS_STICKY:STICKY.NO_L2_L3_FWD_STICKY,
ANA_AC:PS_STICKY:STICKY.IP6_MC_CTRL_FLOOD_STICKY,
ANA_AC:PS_STICKY:STICKY.IP6_MC_DATA_FLOOD_STICKY,
ANA_AC:PS_STICKY:STICKY.IP4_MC_CTRL_FLOOD_STICKY,
ANA_AC:PS_STICKY:STICKY.IP4_MC_DATA_FLOOD_STICKY,
ANA_AC:PS_STICKY:STICKY.L2_MC_FLOOD_STICKY and
ANA_AC:PS_STICKY:STICKY.UC_FLOOD_STICKY

The events can be counted in the port statistics block by setting the corresponding counter mask:ANA_AC:PS_STICKY_MASK:STICKY_MASK

3.21.1.2 Source Lookup

This section describes how to configure source port filtering. The following table lists the applicable registers.

Table 107 • **Source Table Registers Overview**

Target::Register.Field	Description	Replication
ANA_AC:SRC:SRC_CFG	Configures source port filtering	23

Source port filtering for each ingress port can be configured to ensure that frames are not bridged back to the port it was received on (ANA_AC:SRC[0-14]:SRC_CFG).

Ports part of a link aggregation group (LAG) are configured with identical source masks, with all member ports removed (bits corresponding to the member ports are cleared).

If a port is part of a global link aggregation group, a filter is applied to ensure that frames received at this port are not bridged back to any of the ports in the global link aggregation group of which the source port is part (ANA_AC:SRC[15-22]:SRC_CFG).

Local device ports part of a global link aggregation group must be cleared in the global aggregation source mask (via ANA_AC:SRC[15-22]:SRC_CFG).

Source port filtering can be disabled when VCAP IS2 action MASK_MODE is set to 4 (= REDIR_PGID).

Source port filtering per service, required for a hair-pinned service, can be disabled in ANA_L2:ISDX:SERVICE_CTRL.SRC_MASK_DIS.

Source port filtering is not applied when a frame is routed.

3.21.1.3 Aggregation Group Port Selection

This section describes how to configure the aggregation table. The following table lists the applicable registers.

Table 108 • **Aggregation Table Registers Overview**

Target::Register.Field	Description	Replication
ANA_AC:AGGR:AGGR_CFG	Configures aggregation port filtering.	16

The purpose of the aggregation table is to ensure that when a frame is destined for an aggregation group, it is forwarded to exactly one of the group's member ports.

The aggregation code REQ.AGGR_CODE generated in the classifier is used to look up an aggregation mask in the aggregation table. Aggregation mask is configured in ANA_AC:AGGR:AGGR_CFG.

For non-aggregated ports, there is a one-to-one correspondence between logical port (ANA_CL:PORT:PORT_ID_CFG.LPORT_NUM) and physical port, and the aggregation table does not change the forwarding decision.

For aggregated ports, all physical ports in the aggregation group map to the same logical port, and destination entries for a logical port in the PGID table must include all physical ports, which are members

of the aggregation group. Therefore, all but one LAG member port must be removed from the destination port set.

Use of aggregation table can be overruled in ANA_L2 in ANA_L2:ISDX:SERVICE_CTRL.AGGR_REPLACE_ENA.

For more information about link aggregation, see Link Aggregation Code Generation, page 112

3.21.1.4 Global Link Aggregation Forwarding

This section describes how to configure forwarding to a GLAG. The following table lists the applicable registers.

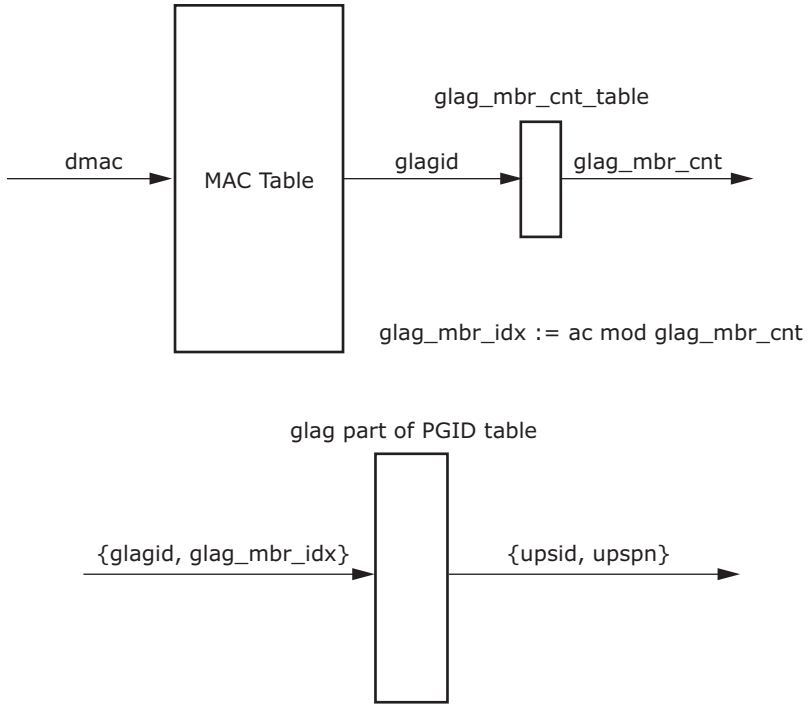
Table 109 • **GLAG Forward Registers Overview**

Target::Register.Field	Description	Replication
ANA_AC::COMMON_VSTAX_CFG	Configures VSTAX handles.	1
ANA_AC:GLAG:MBR_CNT_CFG	Configures number of members per GLAG.	8

The device performs a final port of exit (POE) calculation in ingress device when destination is a global aggregated port. GLAG POE calculation must be enabled (ANA_AC::COMMON_VSTAX_CFG.VSTAX2_GLAG_ENA). When enabled, GLAG forwarding uses a portion of the PGID table (addresses 75 to 138). These addresses must use UPSID, UPSPN encoding by setting the stack interpret bit.

POE calculation, which selects one of the global aggregated as destination port, is shown in the following illustration.

Figure 51 • **GLAG Port of Exit Calculation**



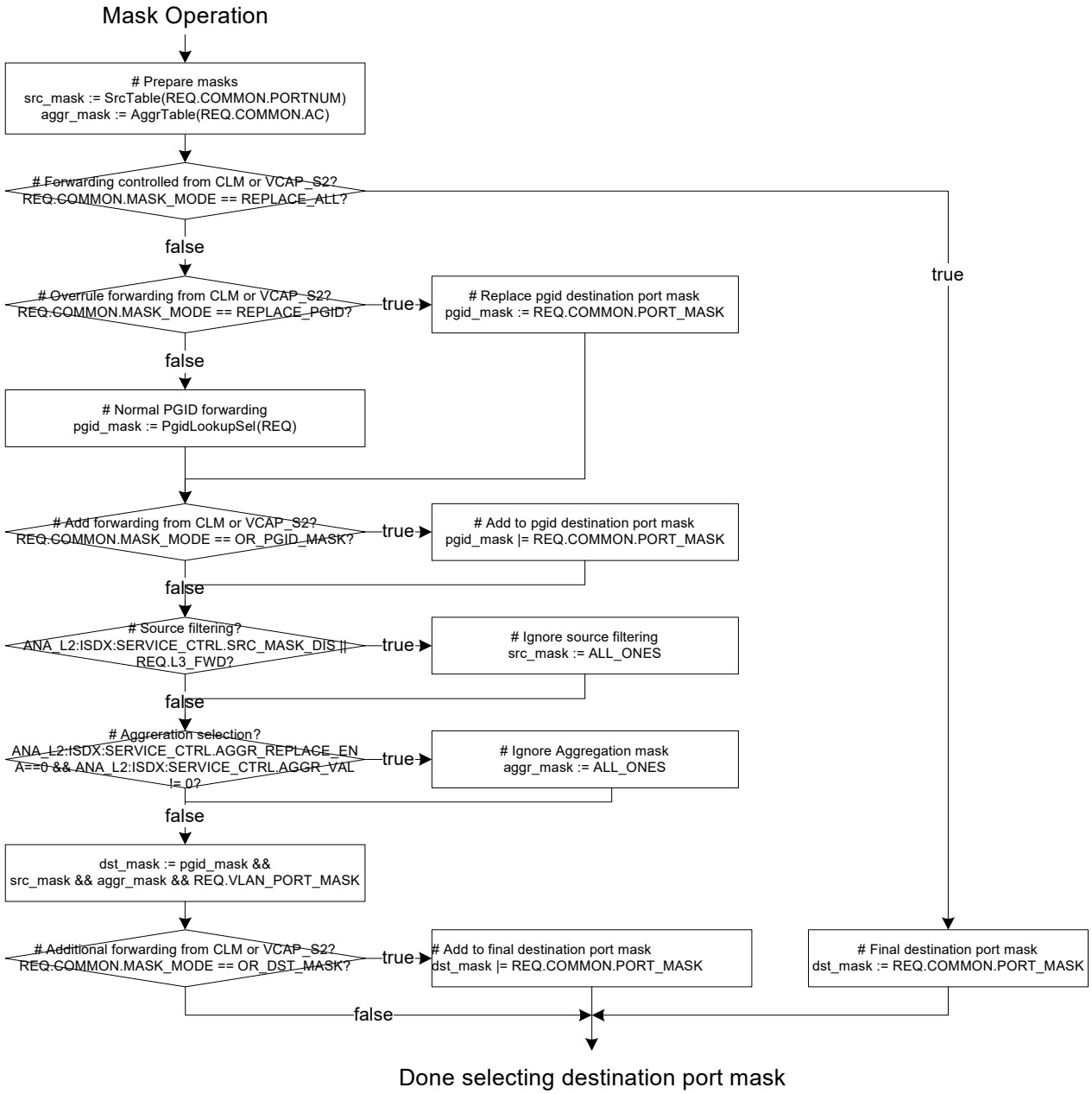
The total number of ports part of a GLAG must be specified in ANA_AC:GLAG:MBR_CNT_CFG.

The corresponding destination must be configured in PGID table for each GLAG member.

3.21.1.5 Port Mask Operation

The final destination port mask is deducted as shown in the following illustration.

Figure 52 • **Port Mask Operation**



3.21.2 Policing

This section describes the functions of the policers. The following table lists the applicable registers.

Table 110 • **Policer Control Registers Overview**

Target::Register.Field	Description	Replication
ANA_AC_POL::POL_ALL_CFG	Configures general control settings.	1
ANA_AC_POL::POL_ACL_CTRL	Configures VCAP policer's mode of operation.	32
ANA_AC_POL::POL_ACL_RATE_CFG	Configures VCAP policer's peak information rate	32
ANA_AC_POL::POL_ACL_THRES_CFG	Configures VCAP policer's burst capacity.	32

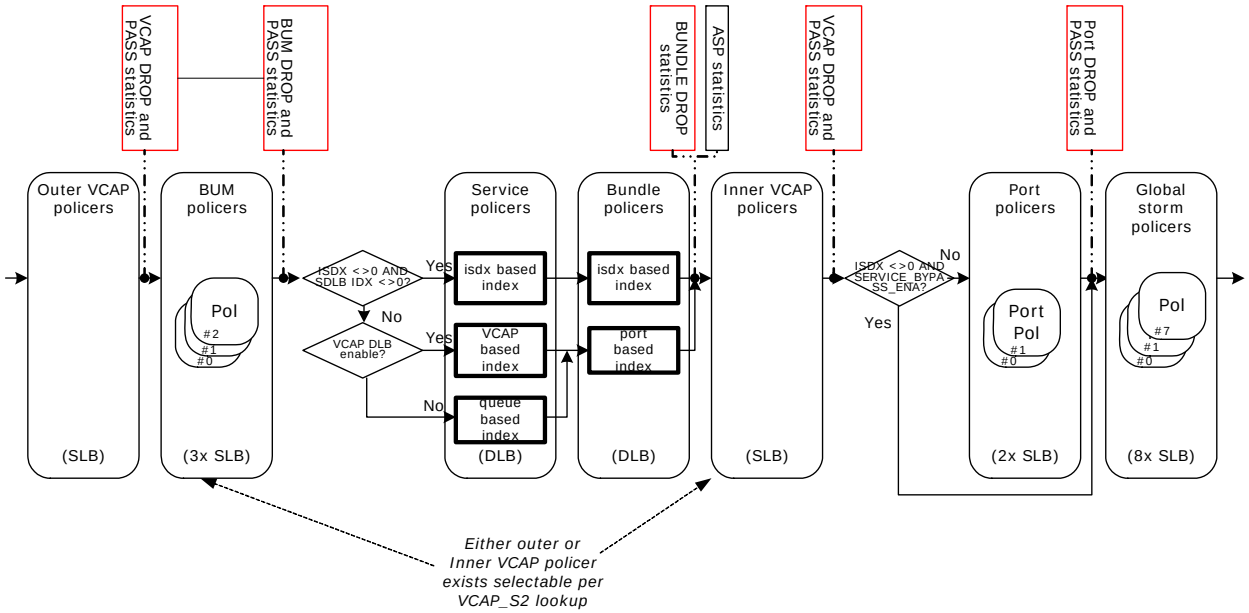
Table 110 • **Policer Control Registers Overview *(continued)***

Target::Register.Field	Description	Replication
ANA_AC_POL::POL_STORM_CTRL	Configures storm policer's mode of operation.	8
ANA_AC_POL::POL_STORM_RATE_CFG	Configures storm policer's peak information rate	8
ANA_AC_POL::POL_STORM_THRES_CFG	Configures Storm policer's burst capacity.	8
ANA_AC_POL::POL_PORT_RATE_CFG	Configures port policer's peak information rate.	Per port per port policer
ANA_AC_POL::POL_PORT_THRES_CFG_0	Configures port policer's burst capacity.	Per port per port policer
ANA_AC_POL::POL_PORT_THRES_CFG_1	Configures port policer's hysteresis when used for flowcontrol.	Per port per port policer
ANA_AC_POL:POL_PORT_CTRL:POL_PORT_CFG	Configures port policer's mode of operation.	Per port per port policer
ANA_AC_POL:POL_PORT_CTRL:POL_PORT_GAP	Configures port policer's gap value used to correct size per frame.	Per port
ANA_AC_POL::POL_STICKY	Captures various policer events for diagnostic purposes.	1
ANA_AC_POL:COMMON_SDLB:DLB_CTRL	Configures common SDLB policer's mode of operation.	1
ANA_AC_POL:COMMON_SDLB:DLB_STICKY	Captures various SDLB policer events for diagnostic purposes.	1
ANA_AC_POL:SDLB:DLB_CFG	Configures SDLB policer's mode of operation.	Per SDLB
ANA_AC_POL:SDLB:LB_CFG	Configures SDLB policer's committed and peak leaky buckets	Per SDLB per leaky bucket

3.21.2.1 Policer Hierarchy

There are multiple levels of policing, which operate in configurable hierarchies, as shown in the following illustration.

Figure 53 • **Policer Hierarchy**



At most, each frame can see one ACL policer. ACL policing can occur as either outer or inner ACL-based. VCAP IS2 action IS_INNER_ACL allows ACL policing to occur before or after service DLB policing and service statistics.

The VCAP, port, broadcast/unicast/multicast (BUM), and storm policers are single leaky bucket (SLB) policers, whereas the service and bundle policers are MEF 10.2 compliant dual leaky bucket (DLB) policers.

The BUM policers group the leaky buckets into sets of three, covering broadcast, unicast, and multicast traffic.

The VCAP, port and storm policers are configurable in steps of 25,040 bits per second, with a minimum of 25,040 bits per second.

The BUM, service, and bundle policers can cover bandwidth in the interval between 0 Gbps and 32 Mbps, with 16 kbps granularity and between 0 Gbps and 10 Gbps with 8 Mbps granularity. The granularity scaling can be configured using the TIMESCALE_VAL configuration parameters.

The service policers can be used as queue policers for non-service traffic when ISDX is zero and ISDX_SDLB index is zero. The queue policer operation is controlled through ANA_L2:COMMON:FWD_CFG.QUEUE_DEFAULT_SDLB_ENA and ANA_L2:COMMON:PORT_DLB_CFG.QUEUE_DLB_IDX.

The bundle policers are only available for services. The index is controlled through ANA_L2:ISDX:MISC_CFG.BDLB_IDX. However, it is possible to use the bundle policers as port policers for non-service traffic when ISDX is zero. The port policer operation is controlled through ANA_L2:COMMON:FWD_CFG.PORT_DEFAULT_BDLB_ENA and ANA_L2:COMMON:PORT_DLB_CFG.PORT_DLB_IDX.

It is possible to specify a pipeline point for the service policers (ANA_L2:ISDX:MISC_CFG.PIPELINE_PT). This can be used to specify where in the processing flow BUM, SDLB, and BDLB policers are active (default set to NONE, which disables any pipeline effect).

The service statistics reflect the decisions made by the service policer. This implies that if a service policer marks a frame yellow, then the yellow service counter is incremented even though the frame might be discarded by one of the subsequent policers.

All policers can be configured to add between –64 and 63 bytes per frame to adjust the counted frame size for inter-frame gap and/or encapsulation.

3.21.2.2 Service and Bundle Dual Leaky Bucket Policing

Each entry in the service and bundle dual leaky bucket tables contains the fields listed in the following table. In this section, “xDLB” is used to reference register groups for both service policers (SDLB) and bundle policers (BDLB).

Table 111 • **Service and Bundle Dual Leaky Bucket Table Entries**

Field	Bits	Description
ANA_AC_POL:xDLB:DLB_CFG.TIMESCALE_VAL	2	Configures DLB policer granularity.
ANA_AC_POL:xDLB:DLB_CFG.COLOR_AWARE_LVL	2	Corresponds to MEF color mode. Specifies the highest DP level that is treated as green (that is, service frames with DP level above COLOR_AWARE_LVL are considered yellow). COLOR_AWARE_LVL == 3 corresponds to color-blind.
ANA_AC_POL:xDLB:DLB_CFG.COUPLING_MODE	1	MEF Coupling Flag (CF). Depending on the setting of COUPLING_MODE, LB_CFG[0] and LB_CFG[1] must be configured as follows: If COUPLING_MODE = 0: LB_CFG[0].RATE_VAL must be configured to MEF CIR. LB_CFG[0].THRES_VAL must be configured to MEF CBS. LB_CFG[1].RATE_VAL must be configured to MEF EIR. LB_CFG[1].THRES_VAL must be configured to MEF EBS. If COUPLING_MODE = 1: LB_CFG[0].RATE_VAL must be configured to MEF CIR. LB_CFG[0].THRES_VAL must be configured to MEF CBS. LB_CFG[1].RATE_VAL must be configured to MEF EIR + MEF CIR. LB_CFG[1].THRES_VAL must be configured to MEF EBS + MEF CBS.
ANA_AC_POL:xDLB:DLB_CFG.CIR_INC_DP_VAL	2	Controls how much drop precedence (DP) is incremented for excess traffic.
ANA_AC_POL:xDLB:DLB_CFG.GAP_VAL	7	Configures the leaky bucket calculation to include or exclude preamble, inter-frame gap and optional encapsulation through configuration.
ANA_AC_POL:xDLB:LB_CFG[0].THRES_VAL	7	Configures committed burst size (CBS).
ANA_AC_POL:xDLB:LB_CFG[1].THRES_VAL	7	Configures DLB’s second threshold. See COUPLING_MODE.
ANA_AC_POL:xDLB:LB_CFG[0].RATE_VAL	11	Configures DLB’s second rate. See COUPLING_MODE.

Each MEF DLB policer supports the following configurations.

- Two rates. Specified in 2048 steps (ANA_AC_POL:xDLB:LB_CFG[0-1].RATE_VAL).
- Two burst sizes. Specified in steps of 2 kilobytes (up to 254 kilobytes) (ANA_AC_POL:xDLB:LB_CFG[0-1].THRES_VAL).
- Color mode: Color-blind or color-aware. Specifies a DP level to separate traffic as green or yellow (ANA_AC_POL:xDLB:DLB_CFG.COLOR_AWARE_LVL). Frames classified with REQ.DP below or equal to COLOR_AWARE_LVL are treated as green. Frames with REQ.DP above COLOR_AWARE_LVL are treated as yellow. A policer is color-blind if configured with a COLOR_AWARE_LVL of 3.
- Coupling flag. Coupled or uncoupled (ANA_AC_POL:xDLB:DLB_CFG.COUPLING_MODE).
- Change DP level. The increase to REQ.DP level when CIR rate is exceeded (ANA_AC_POL:xDLB:DLB_CFG.CIR_INC_DP_VAL).

In addition, the following parameters can also be configured per policer:

- Leaky bucket calculations can be configured to include or exclude preamble, inter-frame gap and an optional encapsulation (ANA_AC_POL:SDLB:DLB_CFG.GAP_VAL).
- Policers can be configured to police CPU traffic, front port forwarded traffic, or both (ANA_AC_POL:xDLB:DLB_CFG.TRAFFIC_TYPE_MASK).

The DLB policers must also be enabled (ANA_AC_POL:COMMON_xDLB:DLB_CTRL.LEAK_ENA and ANA_AC_POL:COMMON_xDLB:DLB_CTRL.DLB_ADD_ENA).

The DLB policer unit size can be adjusted (ANA_AC_POL:COMMON_xDLB: DLB_CTRL.BASE_TICK_CNT) to configure various base units (= smallest rate granularity).

The following DLB policer debug events are available:

- Dropping traffic due to DLB policer can be identified (ANA_AC_POL:POL_ALL_CFG: POL_STICKY.POL_DLB_DROP_STICKY).
- Traffic received without triggering CIR and PIR policing (ANA_AC_POL:COMMON_xDLB:DLB_STICKY.CIR_PIR_OPEN_STICKY).
- Committed information rate exceeded (ANA_AC_POL:COMMON_xDLB:DLB_STICKY.CIR_EXCEEDED_STICKY).
- Peak information rate exceeded (ANA_AC_POL:COMMON_xDLB:DLB_STICKY.PIR_EXCEEDED_STICKY).

3.21.2.3 BUM Policing

Each entry in the BUM single leaky bucket table contains the following fields.

Table 112 • **BUM Single Leaky Bucket Table Entries**

Field	Bits	Description
ANA_AC_POL:BUM_SLB:SLB_CFG.TIMESCALE_VAL	2	Configures policer rate granularity.
ANA_AC_POL:BUM_SLB:SLB_CFG.CIR_INC_DP_VAL	2	Controls how much drop precedence (DP) is incremented for excess traffic.
ANA_AC_POL:BUM_SLB:SLB_CFG.GAP_VAL	7	Configures the leaky bucket calculation to include or exclude preamble, inter-frame gap and optional encapsulation through configuration.
ANA_AC_POL:BUM_SLB:LB_CFG[2:0].THRES_VAL	3 x 7	Configures burst size.
ANA_AC_POL:BUM_SLB:LB_CFG[2:0].RATE_VAL	3 x 11	Configures rate.
ANA_AC_POL:BUM_SLB:SLB_CFG.ENCAP_DATA_DIS	1	Configures if stripped encapsulation data (normalized data) is policed by the policer.
ANA_AC_POL:BUM_SLB:MISC_CFG.FRAME_RATE_ENA	1	Configures frame rate operation.

The BUM policers are indexed through the VLAN table (ANA_L3:VLAN:BUM_CFG.BUM_SLB_IDX). This indexing can be overruled for services through the ISDX table (ANA_L2:ISDX:MISC_CFG.BUM_SLB_IDX and ANA_L2:ISDX:MISC_CFG.BUM_SLB_ENA).

Each BUM policer contains three leaky buckets. Rates and thresholds are configured through ANA_AC_POL:BUM_SLB:LB_CFG).

Traffic for the three BUM leaky buckets is configurable in ANA_AC_POL:COMMON_BUM_SLB:TRAFFIC_MASK_CFG. This provides a flexible allocation of traffic for the policers. For example, it is possible to have BUM configured as:

- Leaky bucket 0: Broadcast traffic (ANA_AC_POL:COMMON_BUM_SLB:TRAFFIC_MASK_CFG[0].TRAFFIC_TYPE_MASK = 1)
- Leaky bucket 1: Unknown unicast traffic (ANA_AC_POL:COMMON_BUM_SLB:TRAFFIC_MASK_CFG[1].TRAFFIC_TYPE_MASK = 4)
- Leaky bucket 2: Unknown multicast traffic (ANA_AC_POL:COMMON_BUM_SLB:TRAFFIC_MASK_CFG[2]. TRAFFIC_TYPE_MASK = 2)

BUM policers can be configured as frame-based policers (ANA_AC_POL:BUM_SLB:MISC_CFG.FRAME_RATE_ENA).

The BUM statistics count on the following events (configured through ANA_AC:STAT_GLOBAL_CFG_BUM:STAT_GLOBAL_EVENT_MASK):

- Bit 0: Count Broadcast traffic discarded by BUM policer
- Bit 1: Count Multicast traffic discarded by BUM policer
- Bit 2: Count Unicast traffic discarded by BUM policer
- Bit 3: Count Broadcast traffic applicable for BUM policer, but not discarded
- Bit 4: Count Multicast traffic applicable for BUM policer, but not discarded
- Bit 5: Count Unicast traffic applicable for BUM policer, but not discarded.

3.21.2.4 ACL Policing

Each ACL policer entry contains the fields listed in the following table.

Table 113 • **ACL Policer Table Entries**

Field	Bits	Description
ACL_TRAFFIC_TYPE_MASK	2	Configures the traffic types to be taken into account by the policer.
FRAME_RATE_ENA	1	Configures frame rate mode for the ACLpolicer, where rates are measured in frames per second instead of bytes per second.
DP_BYPASS_LVL	2	Controls the lowest DP level that is taken into account. That is, traffic with DP below this value is ignored and not policed.
GAP_VALUE	7	Configures the leaky bucket calculation to include or exclude preamble, inter-frame gap and optional encapsulation through configuration.
ACL_THRES	6	Configures ACL policer burst capacity.
ACL_RATE	17	Configures ACL policer rate.

At most, a frame can trigger one ACL policer. ACL policers are enabled by specifying a rate in ANA_AC_POL:POL_ALL_CFG:POL_ACL_RATE_CFG.ACL_RATE and also enabled by a VCAP IS2 hit. For more information, see VCAP IS2, page 61.

The policer burst capacity can be specified with 4K granularity (ANA_AC_POL:POL_ALL_CFG:POL_ACL_THRES_CFG.ACL_THRES).

The following parameters can also be configured per policer.

- The leaky bucket calculation can be configured to include or exclude preamble, inter-frame gap and an optional encapsulation (ANA_AC_POL:POL_ALL_CFG:POL_ACL_CTRL.GAP_VALUE).
- Traffic with DP level below a certain level can be configured to bypass the policers (ANA_AC_POL:POL_ALL_CFG:POL_ACL_CTRL.DP_BYPASS_LVL).
- Each policer can be configured to measure frame rates instead of bit rates (ANA_AC_POL:POL_ALL_CFG:POL_ACL_CTRL.FRAME_RATE_ENA).
- Each policer can be configured to operate on frames towards CPU and/or front ports (ANA_AC_POL:POL_ALL_CFG:POL_ACL_CTRL.ACL_TRAFFIC_TYPE_MASK).

Traffic dropped by an ACL policer can be counted in the ACL statistics block and optionally also in the port statistic block. For more information, see Analyzer Statistics, page 207.

The following ACL policer debug events are available:

- Bypass of policer due to pipeline handling can be identified (ANA_AC_POL:POL_ALL_CFG:POL_STICKY.POL_ACL_PT_BYPASS_STICKY).
- Bypass of policer due to bypass level can be identified (ANA_AC_POL:POL_ALL_CFG:POL_STICKY.POL_ACL_BYPASS_STICKY).
- Dropping of traffic due to ACL policer can be identified (ANA_AC_POL:POL_ALL_CFG:POL_STICKY.POL_ACL_DROP_STICKY).
- Policers that are active but not closed can be identified (ANA_AC_POL:POL_ALL_CFG:POL_STICKY.POL_ACL_ACTIVE_STICKY).

For example, to use ACL policers to limit traffic to CPU only, the following configuration is required:

- Set up a VCAP S2 rule to enable ACL policer 5.
- Configure ACL policer 5 to only allow 100 frames per second towards CPU: # the rate is 10 times the configured value ANA_AC_POL:POL_ALL_CFG:POL_ACL_RATE_CFG[5].ACL_RATE = 10 ANA_AC_POL:POL_ALL_CFG:POL_ACL_CTRL[5].FRAME_RATE_ENA = 1
- Do not accept burst: ANA_AC_POL:POL_ALL_CFG:POL_ACL_THRES_CFG[5].ACL_THRES = 0
- Only police traffic towards CPU: ANA_AC_POL:POL_ALL_CFG:POL_ACL_CTRL[5].ACL_TRAFFIC_TYPE_MASK = 2

3.21.2.5 Priority Policing

The configuration parameters for priority policing are listed in the following table.

Table 114 • **Priority Policer Table Entry**

Field	Description
ANA_L2::FWD.QUEUE_DEFAULT_SDLB_ENA	Enables priority policers for non-service frames.
ANA_L2:PORT:PORT_DLB_CFG.QUEUE_DLB_IDX	Specifies which DLB policers are used for priority policing.

Non-service frames (which ISDX = 0 and ANA_L2:ISDX:DLB_CFG.DLB_IDX = 0) can use the service policers as priority policers. This is enabled in ANA_L2::FWD_CFG.QUEUE_DEFAULT_SDLB_ENA. The frame's ingress port and classified QoS class select the priority policer. The configuration of the priority policer follows the configuration of the service policers. For more information, see Service and Bundle Dual Leaky Bucket Policing, page 201.

Note that a DLB policer index enabled by VCAP IS2 action ACL_MAC[16] takes precedence over the priority policing. In that case, the frame is policed by the VCAP selected policer and not the priority policer.

3.21.2.6 Port Policing

Each port policer entry contains the fields listed in the following table.

Table 115 • **Port Policer Table Entry**

Field	Bits	Description
TRAFFIC_TYPE_MASK	8	Configures the traffic types to be taken into account by the policer.
FRAME_RATE_ENA	1	Configures frame rate mode for the port policer, where rates are measured in frames per second instead of bytes per second.
LIMIT_NONCPU_TRAFFIC_ENA	1	Configures how policing affects traffic towards front ports.
LIMIT_CPU_TRAFFIC_ENA	1	Configures how policing affects traffic towards CPU.
CPU_QU_MASK	8	Configures policing of frames to the individual CPU queues for the port policer (see TRAFFIC_TYPE_MASK).
FC_ENA	1	Configures port policer to trigger flow control.
FC_STATE	1	Current flow control state.
DP_BYPASS_LVL	2	Controls the lowest DP level that is taken into account. That is, traffic with DP below this value is ignored and not policed.
GAP_VALUE	7	Configures the leaky bucket calculation to include or exclude preamble, inter-frame gap and optional encapsulation through configuration.
PORT_THRES0	6	Configures port policer burst capacity.
PORT_THRES1	6	Configures port policer hysteresis size when a port policer is in flow control mode (see FC_ENA).
PORT_RATE	17	Configures port policer rate.

Frames applicable for policing are any frames received by the MAC and forwarded to the classifier. Short frames (less than 148 bytes) with errors, pause frames, or MAC control frames are not forwarded by the MAC and therefore not accounted for in the policers, That is, they are not policed and do not add to the rate measured by the policers.

Port policers are enabled by configuring the traffic type to be policed (ANA_AC_POL:POL_PORT_CTRL:POL_PORT_CFG.TRAFFIC_TYPE_MASK) and specifying a corresponding rate (ANA_AC_POL:POL_PORT_CFG:POL_PORT_RATE_CFG.PORT_RATE).

The policer burst capacity can be specified with 4K granularity (ANA_AC_POL:POL_PORT_CFG:POL_PORT_THRES_CFG_0.PORT_THRES0).

Policing of traffic destined for CPU port or ports can be controlled (in ANA_AC_POL:POL_PORT_CTRL:POL_PORT_CFG.TRAFFIC_TYPE_MASK(7) = 0 and CPU bypass mask in ANA_AC_POL:POL_PORT_CTRL:POL_PORT_CFG.CPU_QU_MASK).

Port policers can individually be configured to affect frames towards CPU ports (ANA_AC_POL:POL_PORT_CTRL:POL_PORT_CFG.LIMIT_CPU_TRAFFIC_ENA) or front ports (ANA_AC_POL:POL_PORT_CTRL:POL_PORT_CFG.LIMIT_NONCPU_TRAFFIC_ENA).

The port policers can be configured for either serial or parallel operation (ANA_AC_POL::POL_ALL_CFG.PORT_POL_IN_PARALLEL_ENA):

- Serial. The policers are checked one after another. If a policer is closed, the frame is discarded and the subsequent policer buckets are not updated with the frame.
- Parallel. The policers are working in parallel independently of each other. Each frame is added to a policer bucket if the policer is open, otherwise the frame is discarded. A frame may be added to one policer although another policer is closed.

The following parameters can also be configured per policer.

- The leaky bucket calculation can be configured to include or exclude preamble, inter-frame gap and an optional encapsulation (ANA_AC_POL:POL_PORT_CTRL:POL_PORT_GAP.GAP_VALUE).
- Each policer can be configured to measure frame rates instead of bit rates (ANA_AC_POL:POL_PORT_CTRL:POL_PORT_CFG.FRAME_RATE_ENA).
- Traffic with DP level below a certain level can be configured to bypass the policers (ANA_AC_POL:POL_PORT_CTRL:POL_PORT_CFG.DP_BYPASS_LVL).

By default, a policer discards frames (to the affected port type: CPU and/or front port or ports) while the policer is closed. A discarded frame is not forwarded to any ports (including the CPU).

However, each port policer has the option to run in flow control where the policer instead of discarding frames instructs the MAC to issue flow control pause frames (ANA_AC_POL:POL_ALL_CFG:POL_PORT_FC_CFG.FC_ENA). When operating in Flow control mode it is possible to specify a hysteresis, which controls when the policer can re-open after having closed (ANA_AC_POL:POL_PORT_CFG:POL_PORT_THRES_CFG_1. PORT_THRES1). The current flow control state is accessible (ANA_AC_POL:POL_ALL_CFG:POL_PORT_FC_CFG.FC_STATE).

Note: Flow control signaling out of ANA_AC must be enabled (ANA_AC_POL::POL_ALL_CFG.PORT_FC_ENA).

To improve fairness between small and large frames being policed by the same policer a hysteresis can be specified for drop mode (ANA_AC_POL:POL_PORT_CFG: POL_PORT_THRES_CFG_1.PORT_THRES1), which controls when the policer can re-open after having closed. By setting it to a value larger than the maximum transmission unit, it can be guaranteed that when the policer opens again, all frames have the same chance of being accepted.

Port policers can be configured operate on logical ports instead of physical ports (ANA_AC_POL::POL_ALL_CFG.LPORT_POLICE_ENA) and thereby allow policing of aggregated port bandwidth.

Traffic dropped by a policer can be counted by the port statistic block.

The following port policer debug events are available.

- Bypass of policer due to pipeline handling can be identified (ANA_AC_POL:POL_ALL_CFG:POL_STICKY.POL_PORT_PT_BYPASS_STICKY).
- Bypass of policer due to DP bypass level can be identified (ANA_AC_POL:POL_ALL_CFG:POL_STICKY.POL_PORT_BYPASS_STICKY).
- Dropping of traffic towards CPU due to policer can be identified (ANA_AC_POL:POL_ALL_CFG:POL_STICKY.POL_PORT_DROP_CPU_STICKY).
- Dropping of traffic towards front port due to policer can be identified (ANA_AC_POL:POL_ALL_CFG:POL_STICKY.POL_PORT_DROP_FWD_STICKY).
- Policers that are active but not closed can be identified per policer (ANA_AC_POL:POL_ALL_CFG:POL_STICKY.POL_PORT_ACTIVE_STICKY).
- Policer triggering flow control can be identified (ANA_AC_POL:POL_ALL_CFG:POL_STICKY.POL_PORT_FC_STICKY).
- Policer leaving flow control state can be identified (ANA_AC_POL:POL_ALL_CFG:POL_STICKY.POL_PORT_FC_CLEAR_STICKY).

3.21.2.7 Storm Policing

Each storm policer entry contains the fields in the following table. The fields are all located within the ANA_AC_POL:POL_ALL_CFG register group.

Table 116 • **Storm Policer Table Entry**

Field	Bits	Description
STORM_TRAFFIC_TYPE_MASK	8	Configures the traffic types to be taken into account by the policer.
STORM_FRAME_RATE_ENA	1	Configures frame rate mode for the ACL policer, where rates are measured in frames per second instead of bytes per second.
STORM_LIMIT_NONCPU_TRAFFIC_ENA	1	Configures how policing affects traffic towards front ports.
STORM_LIMIT_CPU_TRAFFIC_ENA	1	Configures how policing affects traffic towards CPU.
STORM_CPU_QU_MASK	8	Configures policing of frames to the individual CPU queues for the port policer (see TRAFFIC_TYPE_MASK).
STORM_GAP_VALUE	7	Configures the leaky bucket calculation to include or exclude preamble, inter-frame gap and optional encapsulation through configuration
STORM_THRES	6	Configures storm policer burst capacity.
STORM_RATE	17	Configures storm policer rate.

Frames applicable for policing are any frame received by the MAC and forwarded to the classifier. Short frames (less than 148 bytes) with errors, pause frames, or MAC control frames are not forwarded by the MAC and therefore not accounted for in the policers. That is, they are not policed and do not add to the rate measured by the policers.

Storm policers are enabled by configuring the traffic type to be policed (POL_STORM_CTRL.STORM_TRAFFIC_TYPE_MASK) and specifying a corresponding rate (POL_STORM_RATE_CFG.STORM_RATE).

The policer burst capacity can be specified with 4K granularity (POL_STORM_THRES_CFG).

Policing of traffic destined for CPU port or ports can be controlled (POL_STORM_CTRL.STORM_TRAFFIC_TYPE_MASK(7) = 0 and CPU bypass mask in POL_STORM_CTRL.STORM_CPU_QU_MASK).

Storm policers can be configured individually to affect frames towards CPU ports (POL_STORM_CTRL.STORM_LIMIT_CPU_TRAFFIC_ENA) or front ports (POL_STORM_CTRL.STORM_LIMIT_NONCPU_TRAFFIC_ENA).

The following parameters can also be configured per policer.

- Leaky bucket calculation can be configured to include or exclude preamble, inter-frame gap, and an optional encapsulation (POL_ALL_CFG.STORM_GAP_VALUE).
- Each policer can be configured to measure frame rates instead of bit rates (POL_STORM_CTRL.STORM_FRAME_RATE_ENA).

Traffic dropped by a storm policer can be counted by the port statistic block.

The following storm policer debug events are available.

- Dropping of traffic towards CPU due to policer can be identified (POL_STICKY.POL_STORM_DROP_CPU_STICKY).
- Dropping of traffic towards front port due to policer can be identified (POL_STICKY.POL_STORM_DROP_FWD_STICKY).
- Policers that are active but not closed can be identified per policer (POL_STICKY.POL_STORM_ACTIVE_STICKY).

3.21.3 Analyzer Statistics

This section describes how to configure and collect statistics. There are eight statistics counter blocks in the analyzer:

- Port statistics: Four 40-bit counters available are for each port.
- Service statistics: Three 40-bit counters and three 32-bit counters are available for each service.
- Queue statistics: Two 40-bit counters are available for each queue.
- Bundle policer statistics: Two 40-bit counters are available for each bundle policer.
- BUM policer statistics: Six 40-bit counters are available for each BUM policer.
- ACL policer statistics: Two 40-bit counters are available for each ACL policer.
- Ingress router leg statistics: Eight IPv4 40-bit counters and eight IPv6 40-bit counters are available for each ingress router leg.
- Egress router leg statistics: Eight IPv4 40-bit counters and eight IPv6 40-bit counters are available for each egress router leg.

Counters can be set up to count frames or bytes based on configurable criteria. This is described in the following sections.

3.21.3.1 Port Statistics

The following table lists the applicable port statistics registers.

Table 117 • **Analyzer Port Statistics Register Overview**

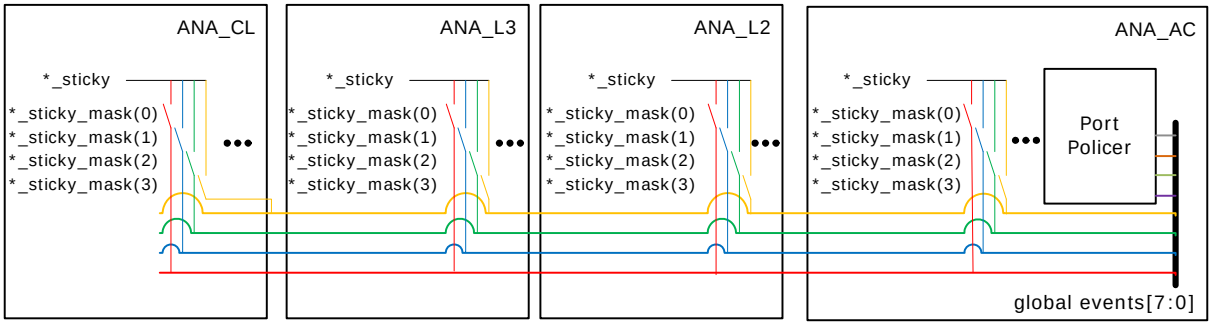
Target::Register.Field	Description	Replication
ANA_AC:STAT_GLOBAL_CFG_PORT. STAT_GLOBAL_EVENT_MASK	Configures global event mask per counter.	4
ANA_AC:STAT_GLOBAL_CFG_PORT:S TAT_RESET.RESET	Initializes or resets all statistic counters and the sticky bits.	1
ANA_AC:STAT_CNT_CFG_PORT.STAT_ CFG. CFG_PRIO_MASK	Configures counting of frames with certain priorities.	per port × 4
ANA_AC:STAT_CNT_CFG_PORT.STAT_ CFG. CFG_CNT_FRM_TYPE	Configures the frame type to be counted.	per port× 4
ANA_AC:STAT_CNT_CFG_PORT.STAT_ CFG. CFG_CNT_BYTE	Configures counting of bytes or frames.	per port × 4
ANA_AC:STAT_CNT_CFG_PORT.STAT_ LSB_CNT	Least significant 32 bits.	per port × 4
ANA_AC:STAT_CNT_CFG_PORT.STAT_ MSB_CNT	Most significant 8 bits.	per port × 4
ANA_AC:STAT_CNT_CFG_PORT.STAT_ EVENTS_STICKY	Sticky bits for counter events.	per port

The port statistics block allows counting of a wide variety of frame events. These are represented by a 12-bit event mask:

- Bits 0–3 represent any stick bit contribution from preceding blocks in ANA.
- Bits 4–7 represent port policer events.
- Bits 8–11 represent storm and ACL policer events and loopback frames.

The propagation of the event mask from the preceding blocks in the analyzer to ANA_AC is shown in the following illustration, except for bits 8–11. See ANA_AC:STAT_GLOBAL_CFG_PORT:STAT_GLOBAL_EVENT_MASK.GLOBAL_EVENT_MASK.

Figure 54 • **Sticky Events Available as Global Events**

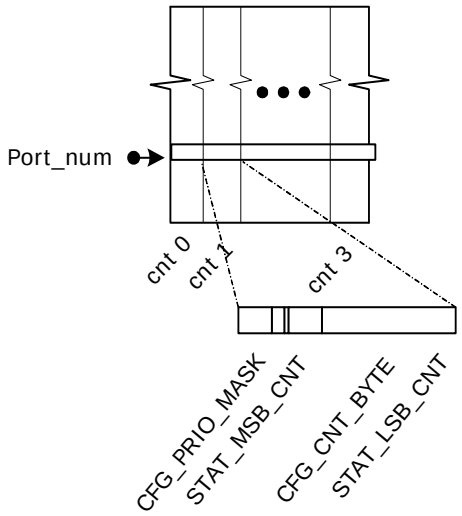


The global events [3:0] are typically allocated for detected control frames, frames dropped due to wrong configuration, frames dropped due to VLAN filtering, and so on.

As an example, a particular event that is normally only of interest during debug, ANA_L2: STICKY.VLAN_IGNORE_STICKY can be configured to be the global event 0 in ANA_L2:STICKY_MASK.VLAN_IGNORE_STICKY_MASK(0).

The following illustration shows the configuration and status available for each port statistics counter.

Figure 55 • **Port Statistics Counters**



Before the statistic counters can be used, they must be initialized to clear all counter values and sticky bits (ANA_AC:STAT_GLOBAL_CFG_PORT:STAT_RESET.RESET).

For each counter the type of frames that are counted must be configured (ANA_AC:STAT_CNT_CFG_PORT.STAT_CFG.CFG_CNT_FRM_TYPE).

Note: Frames without any destination are seen by the port statistics block as aborted.

Each counter must be configured whether frames or bytes are counted. (ANA_AC:STAT_CNT_CFG_PORT.STAT_CFG.CFG_CNT_BYTE).

It is possible to limit counting to certain priorities (ANA_AC:STAT_CNT_CFG_PORT.STAT_CFG.CFG_PRIO_MASK).

Counter events that can trigger counting can be selected among the global events (ANA_AC:STAT_GLOBAL_CFG_PORT. STAT_GLOBAL_EVENT_MASK). There is one common event mask for each of the four counter sets.

Each 40-bit counter consists of an MSB part and an LSB part (ANA_AC:STAT_CNT_CFG_PORT. STAT_MSB_CNT and ANA_AC:STAT_CNT_CFG_PORT.STAT_LSB_CNT). The MSB part of the counter is latched to a shadow register when the LSB part is read. As a result, the LSB part must always be read first, and the MSB part must be read immediately after the LSB part. When writing to the counter, the LSB part must be written first, followed by the MSB part.

The following pseudo code shows the port statistics functionality.

```
for (port_counter_idx = 0; port_counter_idx < 4; port_counter_idx++) {
  if
(STAT_CNT_CFG_PORT[frame.igr_port].STAT_CFG[port_counter_idx].CFG_PRIO_MASK[
frame.prio]) {
    count = FALSE;
    global_event_mask =
STAT_GLOBAL_EVENT_MASK[port_counter_idx].GLOBAL_EVENT_MASK;
   event_match = (STAT_GLOBAL_EVENT_MASK[port_counter_idx].GLOBAL_EVENT_MASK
&
                  frame.event_mask);
    switch
(STAT_CNT_CFG_PORT[frame.igr_port].STAT_CFG[port_counter_idx].CFG_CNT_FRM_TY
PE) {
    case 0x0:
      if (!frame.error && !event_match) count = 1;
      break;
    case 0x1:
      if (!frame.error && event_match) count = 1;
      break;
    case 0x2:
      if (frame.error && event_match) count = 1;
      break;
    case 0x3:
      if (event_match) count = 1;
      break;
    case 0x4:
      if (frame.error && !event_match) count = 1;
      break;
    case 0x5:
      if (frame.error) count = 1;
      break;
    }

    if (count) {
      if
(STAT_CNT_CFG_PORT[frame.igr_port].STAT_CFG[port_counter_idx].CFG_CNT_BYTE) {
        STAT_xSB_CNT[port_counter_idx] += frame.bytes;
      } else {
        STAT_xSB_CNT[port_counter_idx]++;
      }
    }
  }
}
```

Example: To set up port counter 3 to count bytes filtered by VLAN ingress filter, the following configuration is required.

- Enable VLAN drop events as global event 3:
  Set
  ANA_L3:L3_STICKY_MASK:VLAN_MSTP_STICKY_MASK[3].VLAN_IGR_FILTER_STICKY_MAS
  K = 1.
- Initialize counters:
  Set ANA_AC:STAT_GLOBAL_CFG_PORT:STAT_RESET.RESET = 1.
- Set up port counters. For each active Ethernet port:
  Set ANA_AC:STAT_CNT_CFG_PORT[port].STAT_CFG[3].CFG_CNT_FRM_TYPE = 3.
  Set ANA_AC:STAT_CNT_CFG_PORT[port].STAT_CFG[3].CFG_CNT_BYTE = 1.
  Set ANA_AC:STAT_CNT_CFG_PORT[port].STAT_CFG[3].CFG_PRIO_MASK = 0xFF.
- Enable global event 3 as trigger:
  Set ANA_AC:STAT_GLOBAL_CFG_PORT[3]. STAT_GLOBAL_EVENT_MASK = 8.

Example: To set up port counter 0 to count frames dropped by port policer, the following configuration is required.

- Set up port counters. For each active Ethernet port:
  Set ANA_AC:STAT_CNT_CFG_PORT[port].STAT_CFG[0].CFG_CNT_FRM_TYPE = 3.
  Set ANA_AC:STAT_CNT_CFG_PORT[port].STAT_CFG[0].CFG_CNT_BYTE = 0.
  Set ANA_AC:STAT_CNT_CFG_PORT[port].STAT_CFG[0].CFG_PRIO_MASK = 0xFF.
- Enable global event 4 to 7 as trigger:
  Set ANA_AC:STAT_GLOBAL_CFG_PORT[0]. STAT_GLOBAL_EVENT_MASK = 0xF0.

3.21.3.2 Service Statistics

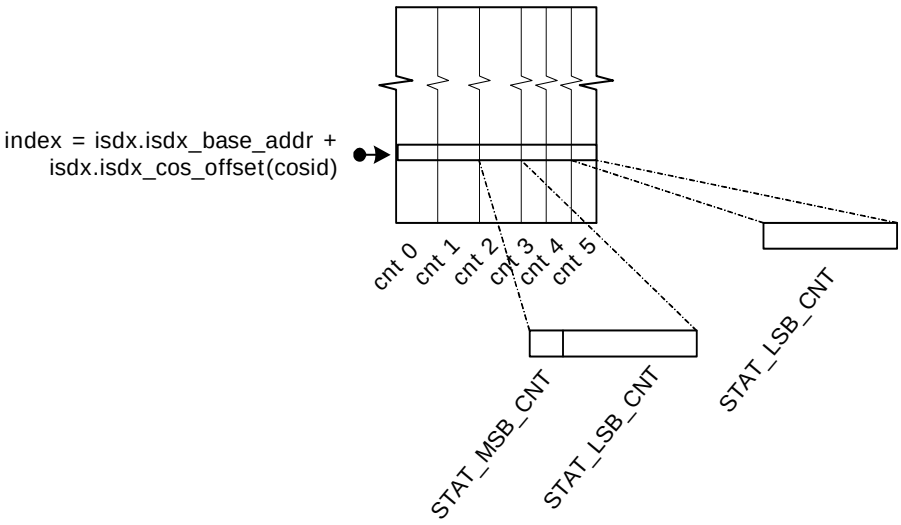
The following table lists the applicable service statistics registers.

Table 118 • **Analyzer Service Statistics Registers Overview**

Target::Register.Field	Description	Replication
ANA_AC:STAT_GLOBAL_CFG_ISDX: STAT_GLOBAL_CFG	Configures counting of bytes or frames per ISDX counter.	6
ANA_AC:STAT_GLOBAL_CFG_ISDX: STAT_GLOBAL_EVENT_MASK	Configures global event mask per ISDX counter.	6
ANA_AC:STAT_GLOBAL_CFG_ISDX: STAT_RESET	Initializes or resets all ISDX statistic counters.	1
ANA_AC_POL:POL_ALL_CFG:POL_ALL_ CFG.USE_SDLB_COLOR_ENA	Configures if service color is determined by DP or by policer color.	1
ANA_AC_POL:POL_ALL_CFG:POL_ALL_ CFG.DP_TO_COLOR_MAP	Configures the mapping between internal DP value and the color used by ISDX counters.	1
ANA_AC:STAT_CNT_CFG_ISDX.STAT_ LSB_CNT	Least significant 32 bits of ISDX or VID counters	per isdx stat entries x 6
ANA_AC:STAT_CNT_CFG_ISDX.STAT_ MSB_CNT	Most significant 8 bits for three lowest per ISDX indexes of ANA_AC:STAT_CNT_CFG_ISDX.STAT_LSB_CNT	per isdx stat entries x 3

The following illustration shows the configuration and status available for each service statistics counter.

Figure 56 • **Service Statistics Counters**



Service statistics can be indexed by one of the following two ways:

- ISDX_BASE_ADDR and ISDX_COS_OFFSET, both configured in ANA_L2:ISDX (shown)
- VID (ANA_AC:PS_COMMON:MISC_CTRL.USE_VID_AS_ISDX_ENA)

Each counter set must be configured whether bytes or frames are counted (ANA_AC:STAT_GLOBAL_CFG_ISDX.CFG_CNT_BYTE).

Counter events that can trigger counting can be selected among the following events:

- Bit 0: Count green traffic
- Bit 1: Count yellow traffic
- Bit 2: Count red traffic
- Bit 3: Count unicast traffic
- Bit 4: Count multicast traffic
- Bit 5: Count flooded traffic

There is one common event mask for each of the six service counter types (ANA_AC:STAT_GLOBAL_CFG_ISDX:STAT_GLOBAL_EVENT_MASK).

The lowest three counter types are 40-bit counters and are intended for counting bytes. The upper counter types are 32-bit counters and are intended for counting frames.

Each 40-bit counter consists of an MSB part and an LSB part (ANA_AC:STAT_CNT_CFG_ISDX. STAT_MSB_CNT ANA_AC:STAT_CNT_CFG_ISDX.STAT_LSB_CNT). The MSB part of the counter is latched to a shadow register when the LSB part is read. As a result, the LSB part must always be read first, and the MSB part must be read immediately after the LSB part. When writing to the counter, the MSB part must be written first, followed by the LSB part.

Example: To set up counting of green, yellow, and red byte and frames per service, use the following configuration.

- Configure byte or frame counting per counter:
  Set ANA_AC:STAT_GLOBAL_CFG_ISDX[0]:STAT_GLOBAL_CFG.CFG_CNT_BYTE = 1 (bytes).
  Set ANA_AC:STAT_GLOBAL_CFG_ISDX[1]:STAT_GLOBAL_CFG.CFG_CNT_BYTE = 1 (bytes).
  Set ANA_AC:STAT_GLOBAL_CFG_ISDX[2]:STAT_GLOBAL_CFG.CFG_CNT_BYTE = 1 (bytes).
  Set ANA_AC:STAT_GLOBAL_CFG_ISDX[3]:STAT_GLOBAL_CFG.CFG_CNT_BYTE = 0 (frames).
  Set ANA_AC:STAT_GLOBAL_CFG_ISDX[4]:STAT_GLOBAL_CFG.CFG_CNT_BYTE = 0 (frames).
  Set ANA_AC:STAT_GLOBAL_CFG_ISDX[5]:STAT_GLOBAL_CFG.CFG_CNT_BYTE = 0 (frames).
- Set up event mask for per counter:
  Set ANA_AC:STAT_GLOBAL_CFG_ISDX[0]:STAT_GLOBAL_EVENT_MASK = 1 (green).
  Set ANA_AC:STAT_GLOBAL_CFG_ISDX[1]:STAT_GLOBAL_EVENT_MASK = 2 (yellow).
  Set ANA_AC:STAT_GLOBAL_CFG_ISDX[2]:STAT_GLOBAL_EVENT_MASK = 4 (red).

Set ANA_AC:STAT_GLOBAL_CFG_ISDX[3]:STAT_GLOBAL_EVENT_MASK = 1 (green).
Set ANA_AC:STAT_GLOBAL_CFG_ISDX[4]:STAT_GLOBAL_EVENT_MASK = 2 (yellow).
Set ANA_AC:STAT_GLOBAL_CFG_ISDX[5]:STAT_GLOBAL_EVENT_MASK = 4 (red).

Example: Set up counting of unicast, multicast, and flooded byte and frames per VID.

- Count per VID:
  Set ANA_AC:PS_COMMON: MISC_CTRL.USE_VID_AS_ISDX_ENA = 1.
- Configure byte or frame counting per counter:
  Set ANA_AC:STAT_GLOBAL_CFG_ISDX[0]: STAT_GLOBAL_CFG.CFG_CNT_BYTE = 1 (bytes).
  ANA_AC:STAT_GLOBAL_CFG_ISDX[1]: STAT_GLOBAL_CFG.CFG_CNT_BYTE = 1 (bytes).
  ANA_AC:STAT_GLOBAL_CFG_ISDX[2]: STAT_GLOBAL_CFG.CFG_CNT_BYTE = 1 (bytes).
  ANA_AC:STAT_GLOBAL_CFG_ISDX[3]: STAT_GLOBAL_CFG.CFG_CNT_BYTE = 1 (frames).
  ANA_AC:STAT_GLOBAL_CFG_ISDX[4]: STAT_GLOBAL_CFG.CFG_CNT_BYTE = 1 (frames).
  ANA_AC:STAT_GLOBAL_CFG_ISDX[5]: STAT_GLOBAL_CFG.CFG_CNT_BYTE = 1 (frames).
- Set up event mask for per counter:
  Set ANA_AC:STAT_GLOBAL_CFG_ISDX[0]:STAT_GLOBAL_EVENT_MASK = 4 (unicast).
  Set ANA_AC:STAT_GLOBAL_CFG_ISDX[1]:STAT_GLOBAL_EVENT_MASK = 8 (multicast).
  Set ANA_AC:STAT_GLOBAL_CFG_ISDX[2]:STAT_GLOBAL_EVENT_MASK = 16 (flood).
  Set ANA_AC:STAT_GLOBAL_CFG_ISDX[3]:STAT_GLOBAL_EVENT_MASK = 4 (unicast).
  Set ANA_AC:STAT_GLOBAL_CFG_ISDX[4]:STAT_GLOBAL_EVENT_MASK = 8 (multicast).
  Set ANA_AC:STAT_GLOBAL_CFG_ISDX[5]:STAT_GLOBAL_EVENT_MASK = 16 (flood).

3.21.3.3 Queue Statistics

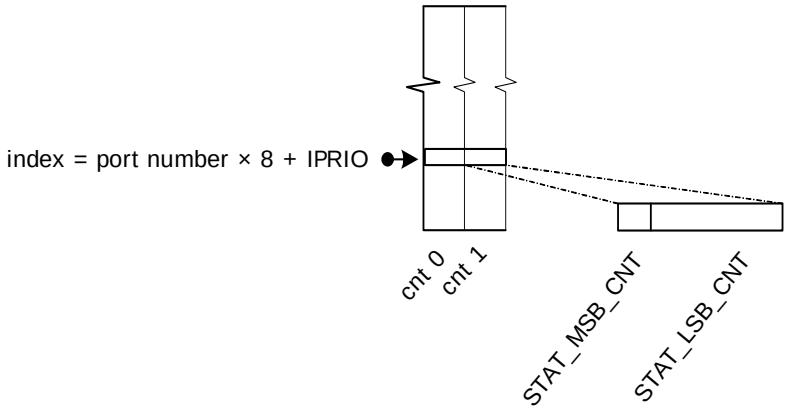
The following table lists the applicable queue statistics registers.

Table 119 • **Queue Statistics Registers Overview**

Target::Register.Field	Description	Replication
ANA_AC:STAT_GLOBAL_CFG_QUEUE: STAT_GLOBAL_CFG	Configures counting of bytes or frames per queue (= port × 8 + iprio).	2
ANA_AC:STAT_GLOBAL_CFG_QUEUE: STAT_GLOBAL_EVENT_MASK	Configures global event mask per queue (= port × 8 + iprio).	2
ANA_AC:STAT_CNT_CFG_QUEUE.STAT_LSB_CNT	Least significant 32 bits of counters per queue.	per queue × 2
ANA_AC:STAT_CNT_CFG_QUEUE.STAT_MSB_CNT	Most significant 8 bits of counters per queue.	per queue × 2

The following illustration shows the configuration and status available for each queue statistics counter.

Figure 57 • **Queue Statistics**



Each counter type must be configured to count either bytes or frames (ANA_AC:STAT_GLOBAL_CFG_QUEUE: STAT_GLOBAL_CFG.CFG_CNT_BYTE).

ACL events that can trigger counting can be selected among the following events:

- Count traffic applicable for queue policer, but not discarded
- Count traffic discarded by queue policer

There is one common event mask for each of the two queue counter types (ANA_AC:STAT_GLOBAL_CFG_QUEUE:STAT_GLOBAL_EVENT_MASK).

Each 40-bit counter consists of an MSB part and an LSB part (ANA_AC:STAT_CNT_CFG_QUEUE.STAT_MSB_CNT and ANA_AC:STAT_CNT_CFG_QUEUE.STAT_LSB_CNT). The MSB part of the counter is latched to a shadow register when the LSB part is read. As a result, the LSB part must always be read first, and the MSB part must be read immediately after the LSB part. When writing to the counter, the MSB part must be written first, followed by the LSB part.

3.21.3.4 Bundle Policer Statistics

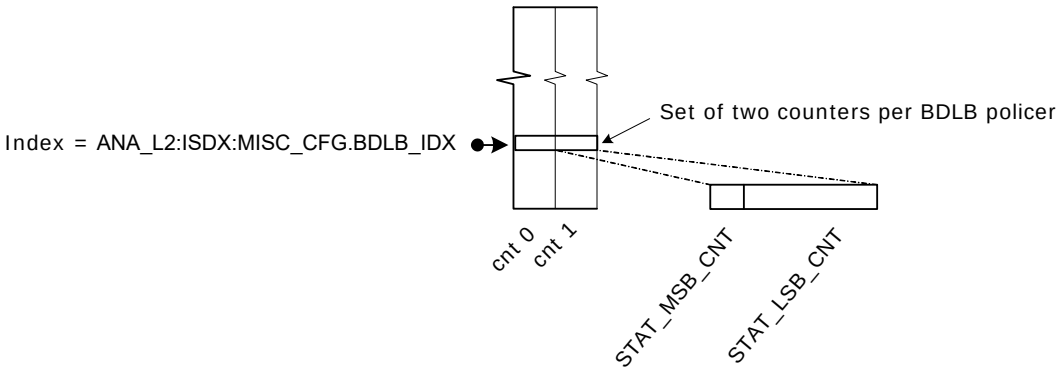
The following table lists the applicable bundle policer statistics registers.

Table 120 • **Bundle Policer Statistics Registers Overview**

Target::Register.Field	Description	Replication
ANA_AC:STAT_GLOBAL_CFG_BDLB:STAT_GLOBAL_CFG	Configures counting bytes or frames for each of the two counters in the counter sets.	2
ANA_AC:STAT_GLOBAL_CFG_BDLB:STAT_GLOBAL_EVENT_MASK	Configures global event mask for each of the two counters in the counter sets.	2
ANA_AC:STAT_CNT_CFG_BDLB.STAT_LSB_CNT	Least significant 32 bits of counters per BDLB policer index.	per BDLB policer index × 2
ANA_AC:STAT_CNT_CFG_BDLB.STAT_MSB_CNT	Most significant 8 bits of counters per BDLB policer index.	per BDLB policer index × 2

The following illustration shows the bundle dual leaky bucket (BDLB) policer statistics. There are two counters available for each BDLB policer.

Figure 58 • **Bundle Policer Statistics**



Each of the two counters in a counter set can be configured to which frames are counted and if either bytes or frames are counted. This configuration is shared among all counter sets.

The BDLB event mask supports counting the following frame types:

- Green traffic
- Yellow traffic
- Red traffic

The event mask is configured in ANA_AC:STAT_GLOBAL_CFG_BDLB:STAT_GLOBAL_EVENT_MASK.

Each 40-bit counter consists of an MSB part and an LSB part (ANA_AC:STAT_CNT_CFG_BDLB.STAT_MSB_CNT and ANA_AC:STAT_CNT_CFG_BDLB.STAT_LSB_CNT). The MSB part of the counter is latched to a shadow register when the LSB part

is read. As a result, the LSB part must always be read first, and the MSB part must be read immediately after the LSB part. When writing to the counter, the MSB part must be written first, followed by the LSB part.

3.21.3.5 Broadcast, Unicast, Multicast (BUM) Policer Statistics

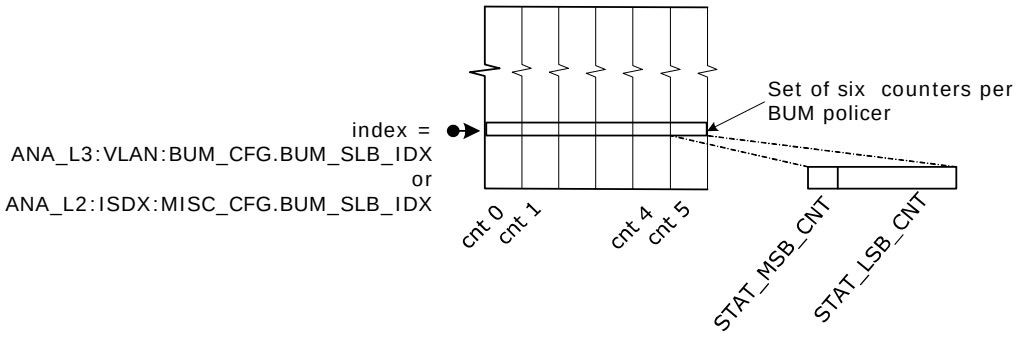
The following table lists the applicable BUM policer statistics registers.

Table 121 • **BUM Policer Statistics Registers Overview**

Target::Register.Field	Description	Replication
ANA_AC:STAT_GLOBAL_CFG_BUM : STAT_GLOBAL_CFG	Configures counting bytes or frames per BUM policer index.	2
ANA_AC:STAT_GLOBAL_CFG_BUM : STAT_GLOBAL_EVENT_MASK	Configures global event mask per BUM policer index.	2
ANA_AC:STAT_CNT_CFG_BUM.STAT_LSB_CNT	Least significant 32 bits of counters per BUM policer index.	per BUM policer index × 2
ANA_AC:STAT_CNT_CFG_BUM.STAT_MSB_CNT	Most significant 8 bits of counters per BUM policer index.	per BUM policer index × 2

The following illustration shows the configuration and status available for each BUM policer.

Figure 59 • **BUM Policer Statistics**



Each of the two counters in a counter set can be configured to which frames are counted and if bytes or frames are counted. This configuration is shared among all counter sets.

The BUM event mask supports counting the following frame types:

- Broadcast traffic discarded by BUM policer
- Multicast traffic discarded by BUM policer
- Unicast traffic discarded by BUM policer
- Broadcast traffic applicable for BUM policer but not discarded
- Multicast traffic applicable for BUM policer but not discarded
- Unicast traffic applicable for BUM policer but not discarded

The event mask is configured in ANA_AC:STAT_GLOBAL_CFG_BUM:STAT_GLOBAL_EVENT_MASK.

Each 40-bit counter consists of an MSB part and an LSB part (ANA_AC:STAT_CNT_CFG_BUM.STAT_MSB_CNT and ANA_AC:STAT_CNT_CFG_BUM.STAT_LSB_CNT). The MSB part of the counter is latched to a shadow register when the LSB part is read. As a result, the LSB part must always be read first, and the MSB part must be read immediately after the LSB part. When writing to the counter, the MSB part must be written first, followed by the LSB part.

3.21.3.6 ACL Policer Statistics

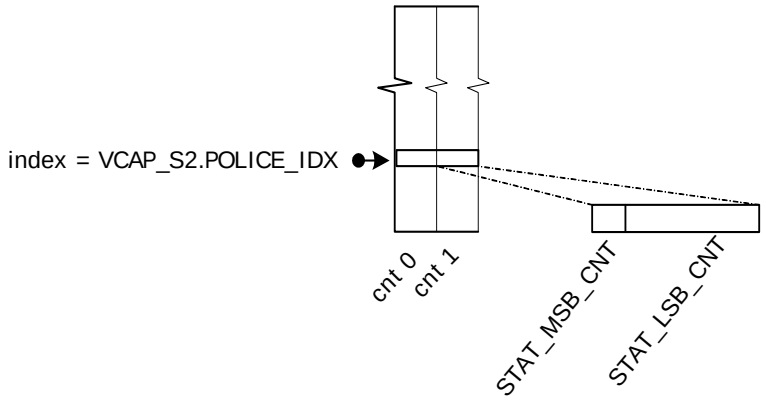
The following table lists the applicable ACL policer statistics registers.

Table 122 • **ACL Policer Statistics Registers Overview**

Target::Register.Field	Description	Replication
ANA_AC:STAT_GLOBAL_CFG_ACL: STAT_GLOBAL_CFG	Configures counting of bytes or frames per ACL policer index.	2
ANA_AC:STAT_GLOBAL_CFG_ACL: STAT_GLOBAL_EVENT_MASK	Configures global event mask per ACL policer index.	2
ANA_AC:STAT_CNT_CFG_ACL.STAT_LSB_CNT	Least significant 32 bits of counters per ACL policer index.	per ACL policer index × 2
ANA_AC:STAT_CNT_CFG_ACL.STAT_MSB_CNT	Most significant 8 bits of counters per ACL policer index.	per ACL policer index × 2

The following illustration shows the configuration and status available for each ACL policer statistics counter.

Figure 60 • **ACL Policer Statistics**



Each counter type must be configured whether bytes or frames are counted (ANA_AC:STAT_GLOBAL_CFG_ACL: STAT_GLOBAL_CFG.CFG_CNT_BYTE).

ACL events that can trigger counting can be selected among the following events. If an ACL policer is triggered by an IS2 action, with IS_INNER_ACL = 1, it is termed “inner”. Otherwise, it is termed “outer”.

- Count CPU traffic applicable for outer ACL policer, but not discarded
- Count front port traffic applicable for outer ACL policer, but not discarded
- Count CPU traffic discarded by outer ACL policer
- Count front port traffic discarded by outer ACL policer
- Count CPU traffic applicable for inner ACL policer, but not discarded
- Count front port traffic applicable for inner ACL policer, but not discarded
- Count CPU traffic discarded by inner ACL policer
- Count front port traffic discarded by inner ACL policer

There is one common event mask for each of the two ACL policer counter types (ANA_AC:STAT_GLOBAL_CFG_ACL:STAT_GLOBAL_EVENT_MASK).

Each 40-bit counter consists of an MSB part and an LSB part (ANA_AC:STAT_CNT_CFG_ACL.STAT_MSB_CNT and ANA_AC:STAT_CNT_CFG_ACL.STAT_LSB_CNT). The MSB part of the counter is latched to a shadow register when the LSB part is read. As a result, the LSB part must always be read first, and the MSB part must be read immediately after the LSB part. When writing to the counter, the MSB part must be written first, followed by the LSB part.

3.21.3.7 Routing Statistics

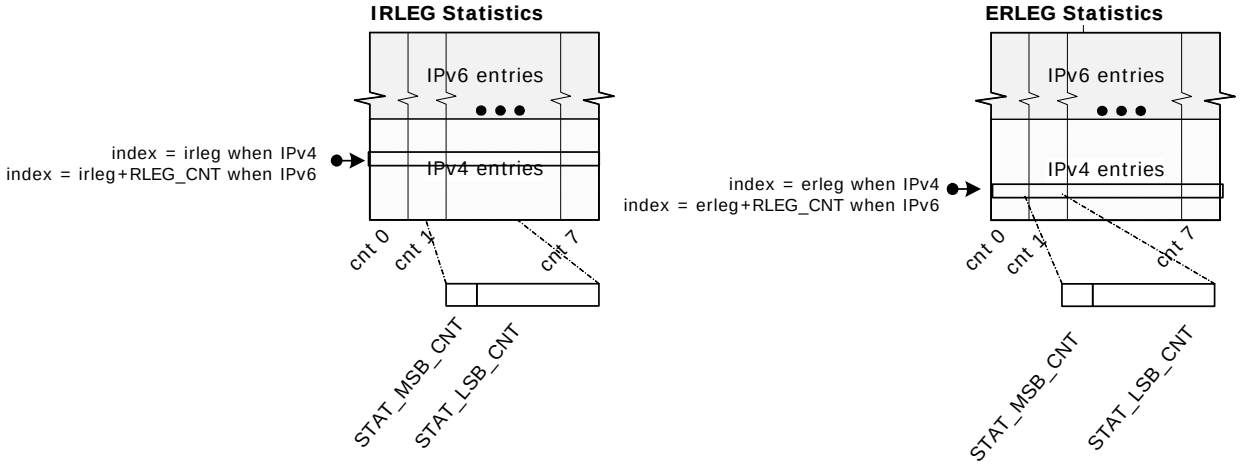
The following table lists the applicable routing statistics registers.

Table 123 • **Analyzer Routing Statistics Registers Overview**

Target::Register.Field	Description	Replication
ANA_AC:STAT_GLOBAL_CFG_IRLEG: STAT_GLOBAL_CFG	Configures counting of bytes or frames per ingress router leg counter.	8
ANA_AC:STAT_GLOBAL_CFG_IRLEG: STAT_GLOBAL_EVENT_MASK	Configures global event mask per ingress router leg counter.	8
ANA_AC:STAT_CNT_CFG_IRLEG.STAT_LSB_CNT	Least significant 32 bits of ingress router leg counters.	Per RLEG per IP_version × 8
ANA_AC:STAT_CNT_CFG_IRLEG.STAT_MSB_CNT	Most significant 8 bits of ingress router leg counters.	Per RLEG per IP_version × 8
ANA_AC:STAT_GLOBAL_CFG_ERLEG: STAT_GLOBAL_CFG	Configures counting of bytes or frames per egress router leg counter.	8
ANA_AC:STAT_GLOBAL_CFG_ERLEG: STAT_GLOBAL_EVENT_MASK	Configures global event mask per egress router leg counter.	8
ANA_AC:STAT_CNT_CFG_ERLEG.STAT_LSB_CNT	Least significant 32 bits of egress router leg counters.	Per RLEG per IP_version × 8
ANA_AC:STAT_CNT_CFG_ERLEG.STAT_MSB_CNT	Most significant 8 bits of egress router leg counters.	Per RLEG per IP_version × 8

The following illustration shows the configuration and status available for ingress router leg and egress router leg statistics counter sets. For this device, RLEG_CNT is 64.

Figure 61 • **Ingress and Egress Routing Statistics per Router Leg per IP Version**



Each counter type must be configured if bytes or frames are counted (ANA_AC:STAT_GLOBAL_CFG_IRLEG: STAT_GLOBAL_CFG.CFG_CNT_BYTE and ANA_AC:STAT_GLOBAL_CFG_ERLEG: STAT_GLOBAL_CFG.CFG_CNT_BYTE)

Ingress router leg counter events that can trigger counting can be selected among the following events:

- acl_discarded - frame applicable for routing but discarded due to VCAP IS2 ingress discard action (first VCAP IS2 lookup with action RT_ENA cleared).
- received IP unicast traffic - frame applicable for routing (hitting a router leg).
- received IP multicast traffic - frame applicable for routing (hitting a router leg).
- unicast_routed - frame is unicast routed.
- multicast_routed - frame is multicast routed

- ip_multicast_rpf_discarded - frame discarded due to failed Reverse Path Forwarding check (frame not received from configured interface)
- ip_TTL_discarded - frame discarded due to TTL <2.

There is one common event mask for each of the eight IRLEG counter types (ANA_AC:STAT_GLOBAL_CFG_IRLEG:STAT_GLOBAL_EVENT_MASK).

Egress router leg counter events that can trigger counting can be selected among the following events:

- acl_discarded - frame applicable for routing but discarded due to VCAP IS2 egress discard action (second VCAP IS2 lookup with action RT_ENA cleared).
- unicast_routed traffic - frame is unicast routed.
- multicast_routed traffic - frame is multicast routed.
- ip_multicast_switched traffic - frame is bridged within ingress VLAN.
- ip_multicast_TTL_discarded - frame discarded due to TTL less than ERLEG configured TTL value.

There is a common event mask for each of the eight ERLEG counter types (ANA_AC:STAT_GLOBAL_CFG_ERLEG:STAT_GLOBAL_EVENT_MASK).

Each 40-bit counter consists of an MSB part (ANA_AC:STAT_CNT_CFG_IRLEG.STAT_MSB_CNT or ANA_AC:STAT_CNT_CFG_ERLEG.STAT_MSB_CNT) and an LSB part (ANA_AC:STAT_CNT_CFG_IRLEG.STAT_LSB_CNT or ANA_AC:STAT_CNT_CFG_ERLEG.STAT_LSB_CNT). The MSB part of the counter is latched to a shadow register when the LSB part is read. As a result, the LSB part must always be read first, and the MSB part must be read immediately after the LSB part. When writing to the counter, the MSB part must be written first, followed by the LSB part.

3.21.4 Analyzer sFlow Sampling

This section describes sFlow sampling. The applicable registers are listed in the following table.

Table 124 • **Analyzer sFlow Registers Overview**

Target::Register.Field	Description	Replication
ANA_AC::PS_COMMON_CFG.SFLOW_ENA	Controls sFlow operation.	1
ANA_AC::PS_COMMON_CFG.SFLOW_SMPL_ID_IN_STAMP_ENA	Configures sFlow sampler ID (port number) as part of stamp sent to CPU.	1
ANA_AC::SFLOW_CFG.SFLOW_CPU_QU	Configures CPU extraction queue for sFlow sampled frames.	1
ANA_AC::SFLOW_CTRL	Configures sFlow samplers (rate and type).	Per port
ANA_AC::SFLOW_CNT	Current sFlow sampler count values.	Per port
ANA_AC:PS_STICKY:STICKY	Various sticky debug events.	1
ANA_AC:PS_STICKY_MASK:STICKY_MASK	Mask to allow debug events to be counted in port stat block.	1

sFlow is a standard for monitoring-high speed switched networks through statistical sampling of incoming and outgoing frames. Each port in the device can be set up as an sFlow agent monitoring the particular link and generating sFlow data. If a frame is sFlow sampled, it is copied to the sFlow CPU extraction queue (ANA_AC::SFLOW_CFG.SFLOW_CPU_QU).

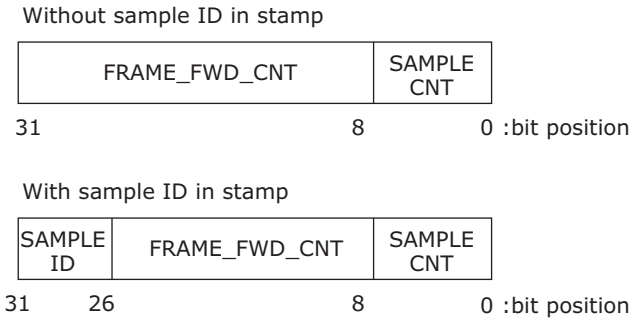
An sFlow agent is configured through ANA_AC::SFLOW_CFG with the following options:

- SFLOW_SAMPLE_RATE specifies the probability that the sampler copies a frame to the CPU. Each frame being candidate for the sampler has the same probability of being sampled. The probability is set in steps of 1/32768.
- SFLOW_DIR_SEL(0) enables incoming frames on the port as candidates for the sampler.
- SFLOW_DIR_SEL(1) enables outgoing frames on the port as candidates for the sampler. When Tx sampling is used, the Sample ID must be set in the sFlow stamp to enable the CPU to determine the sFlow sampler sampled by the frame (ANA_AC::PS_COMMON_CFG.SFLOW_SMPLE_ID_IN_STAMP_ENA).

Rx and Tx sampling can be enabled independently. If both are enabled, all incoming and outgoing traffic on the port is subject to the statistical sampling given by the rate in SFLOW_SAMPLE_RATE.

sFlow sample candidate sent to the CPU is sent with IFH.FWD.SFLOW_MARKING set and with FCS updated as shown in the following illustration.

Figure 62 • **sFlow Stamp Format in FCS**



SAMPLE_CNT is incremented for each sample sent to CPU by the sFlow sampler.

FRAME_FWD_CNT is incremented whenever the frames are applicable for an sFlow sampler.

A variety of sticky events allows debug of the sFlow setup. These events can furthermore be counted in the port statistics block by enabling the corresponding *_MASK. The following events are reported (in ANA_AC:PS_STICKY:STICKY).

- sFlow candidate was found (ANA_AC:PS_STICKY:STICKY.SFLOW_CAND_STICKY).
- sFlow source sample was sent to CPU (ANA_AC:PS_STICKY:STICKY.SFLOW_SRC_SAMPLE_STICKY).
- sFlow destination sample was sent to CPU (ANA_AC:PS_STICKY:STICKY.SFLOW_DST_SAMPLE_STICKY).

Current sample count and sample candidate count numbers are available (ANA_AC::SFLOW_CNT.SFLOW_SAMPLE_CNT and ANA_AC::SFLOW_CNT.SFLOW_FRAME_FWD_CNT).

These events can all be counted in the statistics block by enabling the corresponding *STICKY_MASK found in register group ANA_AC:PS_STICKY_MASK:STICKY_MASK[x]. (ANA_AC:PS_STICKY_MASK:STICKY_MASK.SFLOW_CAND_STICKY_MASK, ANA_AC:PS_STICKY_MASK:STICKY_MASK.SFLOW_DST_SAMPLE_STICKY_MASK, ANA_AC:PS_STICKY_MASK:STICKY_MASK.SFLOW_SRC_SAMPLE_STICKY_MASK and ANA_AC:PS_STICKY_MASK:STICKY_MASK.SFLOW_SAMPLE_STICKY_MASK).

3.21.5 Mirroring

This section describes how to configure mirroring. The following table lists the applicable registers.

Table 125 • **ANA_AC Mirror Registers Overview**

Target::Register.Field	Description	Replication
ANA_AC:MIRROR_PROBE:PROBE_CFG.PROBE_DIRECTION	Configures whether to probe ingress traffic, egress traffic, or both.	3
ANA_AC:MIRROR_PROBE:PROBE_CFG.PROBE_MAC_MODE	Configures MAC address filtering i.e. only traffic to and/or from hosts known in to MAC table with the mirror bit set are mirrored.	3
ANA_AC:MIRROR_PROBE:PROBE_CFG.PROBE_VLAN_MODE	Configures VLAN filtering. That is, only traffic with a specific VID or with VLAN entry mirror bit set are mirrored.	3
ANA_AC:MIRROR_PROBE:PROBE_CFG.PROBE_VID	Configures VID when PROBE_VLAN_MODE is set to probe VID.	3
ANA_AC:MIRROR_PROBE:PROBE_CFG.PROBE_CPU_SET	Configures mirroring of traffic to specific CPU ports.	3

Table 125 • **ANA_AC Mirror Registers Overview *(continued)***

Target::Register.Field	Description	Replication
ANA_AC:MIRROR_PROBE:PROBE_CFG.PROBE_PHYS_RX_PORT	Configures if Rx mirroring mirrors physical or masqueraded port.	3
ANA_AC:MIRROR_PROBE:PROBE_CFG.PROBE_RX_CPU_AND_VD	Configures Rx mirror mask for CPU ports and VD ports.	3
ANA_AC:MIRROR_PROBE:PROBE_PORT_CFG	Configures mirroring of the ingress ports for PROBE_DIRECTION = RX_MIRROR and/or the set of egress ports for PROBE_DIRECTION = TX_MIRROR.	3
ANA_AC:MIRROR_PROBE:PROBE_PORT_CFG1	See PROBE_PORT_CFG.	3
QFWD:SYSTEM:FRAME_COPY_CFG[9-11]	Configures forwarding options per probe.	3

To debug network problems, selected traffic can be mirrored using configurable probes, to a mirror port where a frame analyzer can be attached to analyze the frame flow.

The device has three independent mirroring probes. Each probe can be configured with a separate set of filtering conditions that must be fulfilled before traffic is mirrored. If multiple filtering conditions are enabled for a given probe they must all be fulfilled before mirroring occurs. For example, mirror probe 1 can be set up to only Rx mirror traffic from a given host in a specific VLAN, whereas probe 2 can be set up to mirror frames matching criteria in VCAP IS2. If a frame is mirrored by more than one mirror probe, the highest numbered probe is selected. This implies that only one mirror copy is made per frame, even if multiple probes are active.

The following traffic mirror conditions can be configured per probe:

- All frames received on a given port, also known as ingress mirroring (enabled in ANA_AC:MIRROR_PROBE[x].PROBE_CFG.PROBE_DIRECTION(1) = 1 and ports to be RX mirrored set in ANA_AC:MIRROR_PROBE[x].PROBE_PORT_CFG)
- All frames transmitted on a given port, also known as egress mirroring (enabled in ANA_AC:MIRROR_PROBE[x].PROBE_CFG.PROBE_DIRECTION(0) = 1 and ports to be TX mirrored set in ANA_AC:MIRROR_PROBE[x].PROBE_PORT_CFG)
- All frames sent to a specific CPU port (may be useful for software debugging) (enabled in ANA_AC:MIRROR_PROBE[x].PROBE_CFG.PROBE_DIRECTION(0) = 1 and CPU ports to be mirrored set in ANA_AC:MIRROR_PROBE[x].PROBE_CFG.PROBE_CPU_SET)
- All frames classified to specific a VID (enabled in ANA_AC:MIRROR_PROBE[x]. PROBE_CFG. PROBE_VLAN_MODE == 2 and VID set in ANA_AC:MIRROR_PROBE[x].PROBE_CFG.PROBE_VID)
- All frames classified to VLANs with VLAN->VLAN_MIRROR_ENA set (enabled in ANA_AC:MIRROR_PROBE[x].PROBE_CFG.PROBE_VLAN_MODE== 1).
- All frames received from a known station with MAC->MIRROR set (enabled in ANA_AC:MIRROR_PROBE[x].PROBE_CFG.PROBE_MAC_MODE(1)= 1).
- All frames send to a known station with MAC->MIRROR set (enabled in ANA_AC:MIRROR_PROBE[x].PROBE_CFG. PROBE_MAC_MODE(0) = 1). This can also be flooded traffic (enabled via ANA_L2::FWD_CFG.FLOOD_MIRROR_ENA).
- Frames selected through configured VCAP entries (enabled by VCAP IS2 action MIRROR_PROBE). For such frames, the only other applicable mirror criteria is PROBE_DIRECTION.
- All frames received from CPU, also known as CPU ingress mirroring (enabled in ANA_AC:MIRROR_PROBE[x].PROBE_CFG.PROBE_DIRECTION(1) = 1 and CPU ports to be RX mirrored set in ANA_AC:MIRROR_PROBE[x].PROBE_RX_CPU_AND_VD).

The mirror port configured per probe (QFWD:SYSTEM:FRAME_COPY_CFG[8-11]) can be any port on the device, including the CPU.

Rx-mirrored traffic sent to the mirror port is sent as received optionally with an additional Q-tag (enabled in REW::MIRROR_PROBE_CFG.REMOTE_MIRROR_CFG).

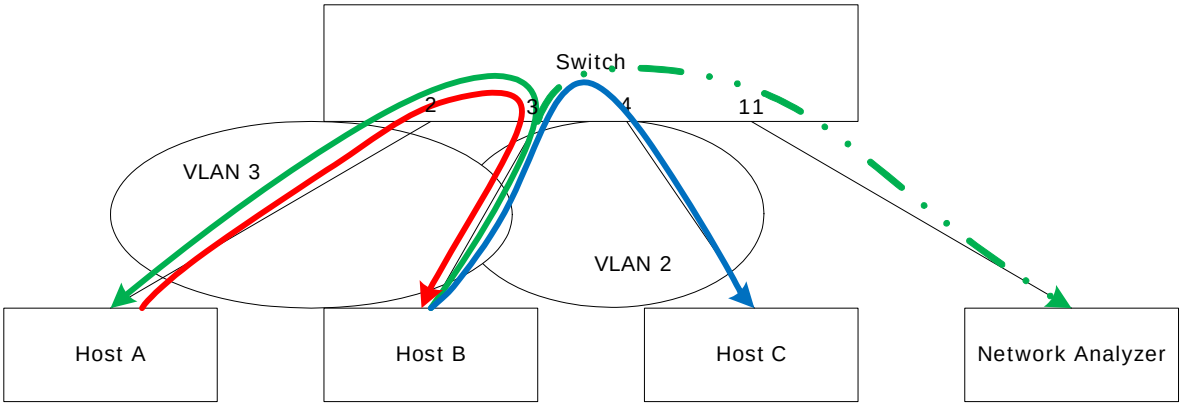
Tx-mirrored traffic sent to the mirror port is modified according to a configured Tx port (REW:COMMON:MIRROR_PROBE_CFG.MIRROR_TX_PORT) optionally with an additional Q-tag (enabled in REW::MIRROR_PROBE_CFG.REMOTE_MIRROR_CFG).

For information about possible rewriter options for mirrored frames, see Mirror Frames, page 351.

Example: Ingress port mirroring in specific VLAN

All traffic only in VLAN 3 from host B on port 3 (the probe port) is mirrored to port 11 (the mirror port) using probe 1, as shown in the following illustration.

Figure 63 • **Ingress Mirroring in Specific VLAN**



The following configuration is required:

- Enable RX mirroring on port 3:
  ANA_AC:MIRROR_PROBE[1].PROBE_CFG.PROBE_DIRECTION = 2 (= Rx only)
  ANA_AC:MIRROR_PROBE[1].PROBE_PORT_CFG = 0x8 (port 3)
- Enable VID filtering:
  ANA_AC:MIRROR_PROBE[1].PROBE_CFG.PROBE_VLAN_MODE = 2
  ANA_AC:MIRROR_PROBE[1].PROBE_CFG. PROBE_VID = 3
- Setup mirror port: QFWD:SYSTEM:FRAME_COPY_CFG[10].FRMC_PORT_VAL 11

3.22 Versatile OAM Processor (VOP)

This section provides information about the functionality implemented by the Microsemi's Versatile OAM processor (VOP). The VOP can process frames at wire-speed.

3.22.1 VOP Blocks

The VOP implements the following functional blocks.

- Versatile OAM endpoints (VOE)
- Loss of continuity controller (LOCC)
- Auto Hit-Me-Once controller (HMOC)
- Interrupt controller
- Port count-all Rx/Tx counters

3.22.1.1 Versatile OAM Endpoint (VOE)

The VOP implements 203 Versatile OAM endpoints (VOEs). One VOE handles a bidirectional OAM flow (both Rx and Tx) including OAM PDU updates, statistics, and counters.

The 203 VOEs are split into the following two groups:

- 192 service VOEs
- 11 port VOEs

There is one port VOE per front port and a one-to-one mapping between the port VOEs and the front ports. Port VOEs operate on the physical port and are unaware of any port aggregation. Service VOEs and port VOEs can only reside on a single front port.

VOEs are configured for either Down-MEP or UP-MEP function:

A VOE configured for Down-MEP operation is denoted a Down-VOE, and a VOE configured for Up-MEP operation is denoted a Up-VOE. Service VOEs can operate as either Down-VOEs or Up-VOEs. Port VOEs can only operate as Down-VOEs.

VOEs are configured to support one of the following OAM types:

- Ethernet OAM (Y.1731, G.8021 and 802.1ag)
- G.8113.1 (Y.1732 based MPLS-TP OAM)
- MPLS-TP (BFD: RFC-6428)
  Other MPLS-TP G-ACH channel types can be counted and extracted to CPU.

A VOE configured for Ethernet functionality is denoted an Ethernet VOE, and a VOE configured for MPLS-TP functionality is denoted an MPLS-TP VOE.

3.22.1.2 Loss of Continuity Controller (LOCC)

In addition to implementing the VOEs, the VOP implements a loss of continuity controller (LOCC). The LOCC is used commonly by all VOEs to detect loss of continuity (LOC). The LOCC is used both for Ethernet VOEs and MPLS-TP VOEs.

3.22.1.3 Hit-Me-Once Controller (HMOC)

The Hit-Me-Once controller (HMOC) supports asserting VOE extraction bits at programmable intervals for selected frame types. This allows periodic extraction of frames to the CPU, without CPU intervention. This functionality is referred to as auto HMO and is supported only for Ethernet VOEs.

3.22.1.4 Interrupt Controller

The VOP includes an interrupt controller for controlling the individual VOE interrupts. The interrupt controller generates a single VOP interrupt connected as OAM_VOP to the VCore-III interrupt controller.

3.22.1.5 Port Count-All Rx/Tx Counters

For every physical port, the VOP implements a set of counters that count all the bytes and frames that pass the port VOE in the Rx/Tx direction. The counters are maintained per COSID. The COSID for any frame is identical to the COSID as configured for the port VOE on the port. These counters are not OAM-related.

3.22.2 Versatile OAM Endpoint (VOE) Functions

The Versatile OAM Endpoint (VOE) relies on the frame classification done by the analyzer to distinguish between different frame types. The VOE recognizes the following IFH frame types in IFH.DST.ENCAP.PDU_TYPE:

- OAM_Y1731 (Ethernet OAM)
- OAM_Y1731 (G.8113.1 MPLS-TP OAM)
- OAM_MPLS_TP (G.8113.2 MPLS-TP OAM)
- SAM_SEQ (non-OAM SAM sequence numbering)

Because of the similarity between G.8113.1 OAM PDUs and Y.1731 OAM PDUs, the processing of G.8113.1 OAM PDUs is done by a VOE configured as an Ethernet VOE and enabled for G.8113.1 processing.

In the following sections, all functionality described for Ethernet VOEs also applies to G.8113.1 OAM PDUs, unless explicitly stated.

Frames classified to other frame types than the above are processed as data frames. Data frames are not analyzed further by the VOE. Data frames are counted as part of the Frame Loss Measurement (F-LM) counters and optionally discarded.

Frames classified as OAM frame types are analyzed by the VOE and are subject to OAM PDU specific processing. The VOE processing of OAM PDUs include the following functions:

- OAM PDU recognition
- OAM PDU validation
- OAM statistics and PDU counters

- OAM loss measurement counters (Ethernet only)
- Sticky bit updating
- OAM PDU frame modification
- OAM PDU extraction, looping, or termination

Processing of the OAM PDU types supported by the VOE is enabled individually per OAM PDU type. When disabled, the VOE does not update the OAM PDU, and it does not OAM PDU specific validation. The VOE statistics and frame extraction are enabled independently of the processing of the OAM PDU types. When enabled, the VOE updates the OAM PDU in addition to Rx PDU validation, statistics and frame extraction. In scenarios where the CPU performs the OAM PDU updates, the VOE updating of the OAM PDU can be disabled.

3.22.3 Supported OAM PDUs

The VOE implements dedicated functions for selected OAM PDUs. OAM PDUs, for which there is no dedicated support, can be extracted to the CPU for further processing. The supported OAM PDUs depends on if the VOE is configured for Ethernet OAM or MPLS-TP OAM.

3.22.3.1 Ethernet OAM (Y.1731 and G.8113.1 OAM)

Ethernet VOEs include support for the following Y.1731 OAM PDUs. Ethernet OAM PDUs not listed can be classified as Generic Opcodes with dedicated statistics and extraction to the CPU.

- TST
- LBM/LBR
- LMM/LMR
- DMM/DMR
- 1DM
- CCM
- CCM-LM
- SLM/SLR
- 1SL
- LTM/LTR (extraction only, no processing)

The OAM PDUs can be processed below any of the Ethernet encapsulations that are supported by the device ranging from untagged frames with no MPLS encapsulation to MPLS frames with two MPLS link layer VLAN tags, three label stack entries, a control word, and three customer VLAN tags.

The Ethernet VOE can insert and check sequence numbers in non-OAM data frames. This is known as SAM sequence numbering (SAM_SEQ) and is intended for SAM testing.

3.22.3.2 MPLS-TP OAM (G.8113.2 OAM)

MPLS-TP VOEs include support for the following MPLS-TP G-ACH channel types.

- BFD-CC (ACH Channel Type: 0x0022)
- BFD-CC (ACH Channel Type: 0x0007)
- BFD-CV (ACH Channel Type: 0x0023)

The OAM PDUs can be processed below any of the MPLS encapsulations supported by the device, including MPLS-TP label stack of three labels and one reserved label.

MPLS-TP G-ACH Channel Types not known to the MPLS-TP VOE can be extracted to the CPU.

3.22.4 VOE Locations

The VOP processes frames due to a VOE lookup done in the analyzer and another VOE lookup done in the rewriter. To describe the processing done by the VOE in the analyzer and rewriter the following terms

are defined. The terms ingress and egress are used with respect to the flow through the switch-core while the terms Rx and Tx are used with respect to the VOE.

Table 126 • **VOE Terminology**

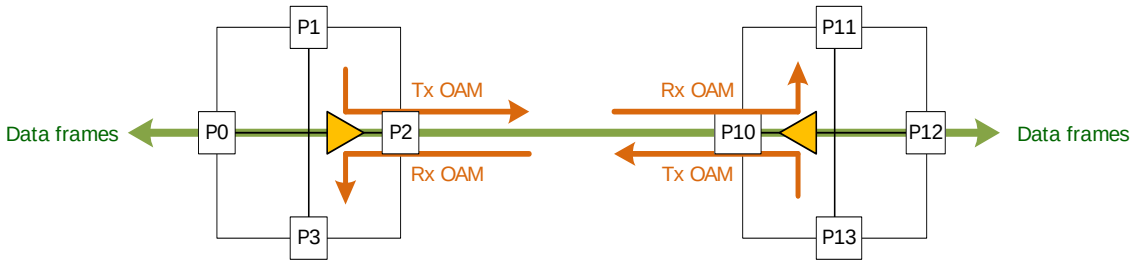
Terms	Description	Down-VOE Lookup	Up-VOE Lookup
Ingress VOE lookup	Denotes the VOE lookup done by the analyzer. The ingress VOE lookup processes ingress OAM and data frames. Ingress VOE lookups apply to both Down-VOEs and Up-VOEs.	ANA	ANA
Egress VOE lookup	Denotes the VOE lookup done by the rewriter. The egress VOE lookup processes egress OAM and data frames. Egress VOE lookups apply to both Down-VOEs and Up-VOEs.	Rewriter	Rewriter
Rx OAM Rx frame	Denotes an OAM or data frame being processed by the VOE and flowing in the same direction as the VOE receives OAM PDUs from the peer MEP.	Analyzer	Rewriter
Tx OAM Tx frame	Denotes an OAM or data frame being processed by the VOE and flowing in the same direction as the VOE transmits its own OAM PDUs towards the peer MEP.	Rewriter	Analyzer
VOE Rx lookup	The VOE lookup processing Rx OAM and data frames received from the peer MEP.	Analyzer	Rewriter
VOE Tx lookup	The VOE lookup processing Tx OAM and data frames forwarded towards the peer MEP.	Rewriter	Analyzer

In the following subsections, the terms in the previous table are described in the context of Down-VOEs and Up-VOEs.

3.22.4.1 VOE: Down-MEP Location

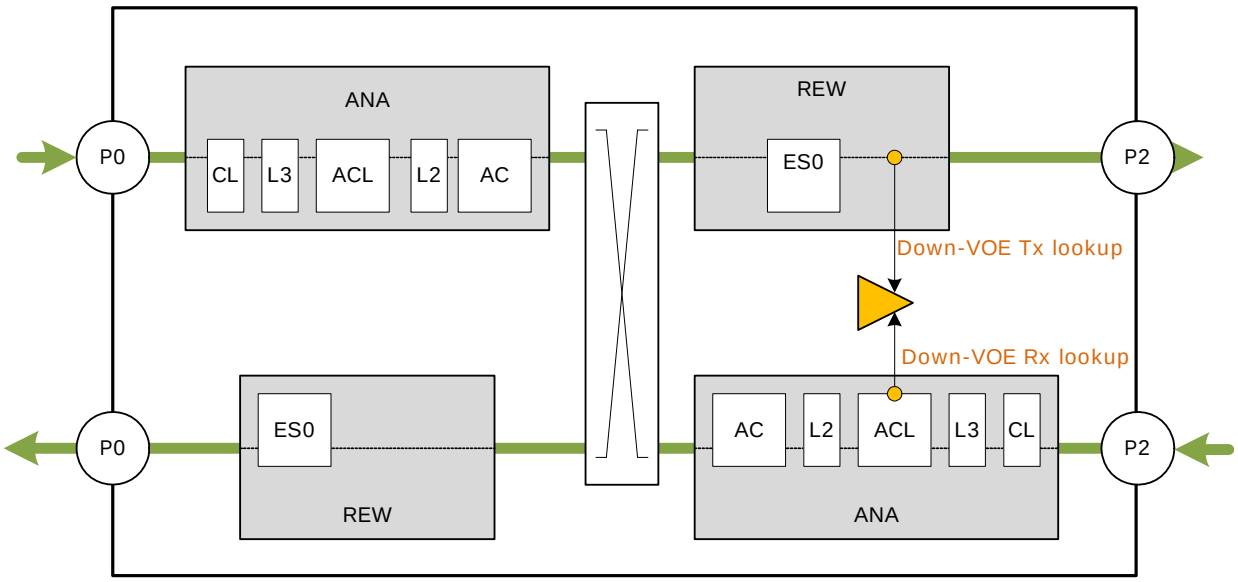
The following illustration shows general location of a Down-MEP in a switch. It shows the bidirectional OAM PDU flow between two Down-MEPs (shown using orange) and the bidirectional data frame flow being monitored by the two Down-MEPs (shown using green). The Down-MEP injects the Tx OAM in the egress frame flow after the queue system towards the front port. This defines the point of the Down-VOE Tx lookup. Likewise Rx OAM frames are extracted from the ingress frame flow directly from the front port before the queue system. This defines the point of the Down-VOE Rx lookup.

Figure 64 • **Down-MEP Location**



The following illustration shows the detailed frame flow between ports P0 and P2, depicted in the previous illustration. The frame flow from left to right is illustrated at the top of the illustration, and the frame flow from right to left is shown at the bottom.

Figure 65 • **Down-VOE Location**



The illustration shows a Down-VOE, which resides on port 2, where the following applies:

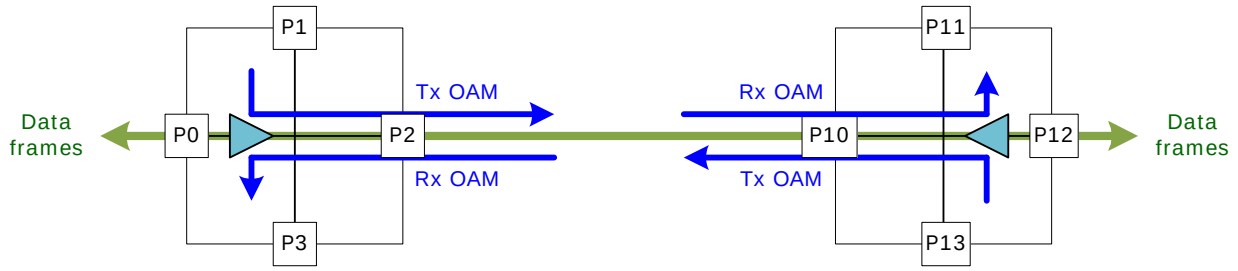
- The Down-VOE Tx lookup is performed by the rewriter and it processes egress OAM and data frames.
- The Down-VOE Rx lookup is performed by the analyzer and it processes ingress OAM and data frames.

A single Down-VOE processes both ingress (Rx) and egress (Tx) frames.

3.22.4.2 VOE: Up-MEP Location

The following illustration shows the general location of an Up-MEP in a switch.

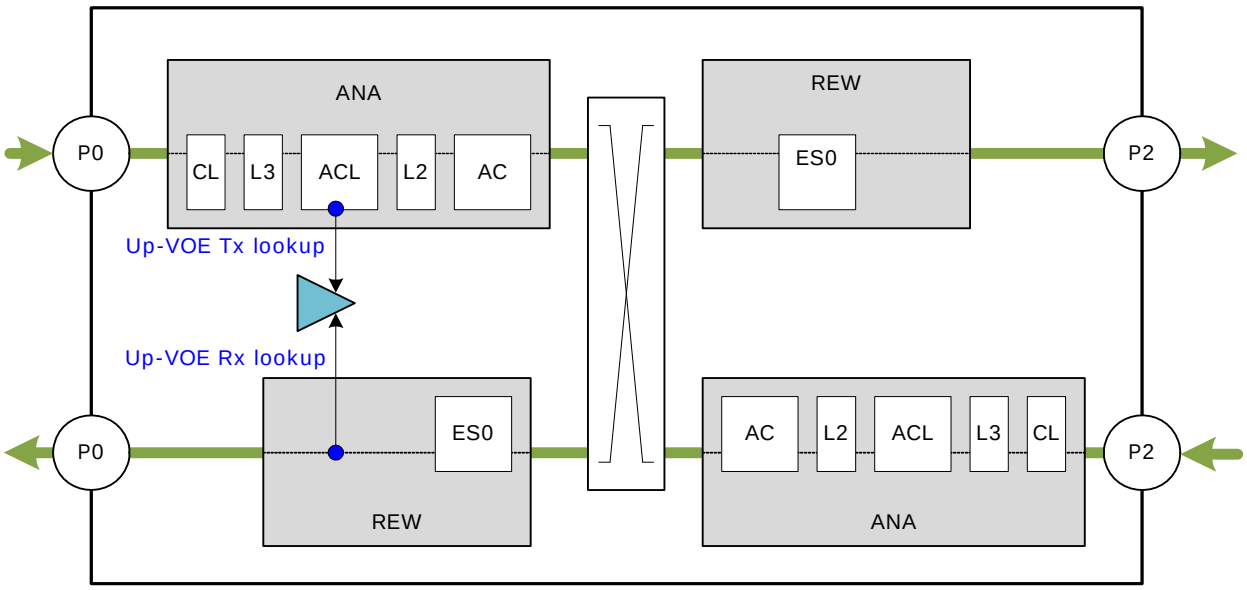
Figure 66 • **Up-MEP Location**



It shows the bidirectional OAM PDU flow between two Up-MEPs (shown using blue) and the bidirectional data frame flow being monitored by the two Up-MEPs (shown using green). The Up-MEP injects the Tx OAM in the ingress frame flow after the front port towards the queue system. This defines the point of the Up-VOE Tx lookup. Likewise Rx OAM frames are extracted from the egress frame flow directly from the queue system before reaching the front port. This defines the point of the Up-VOE Rx lookup.

The following illustration shows the detailed frame flow between ports P0 and P2 depicted in the previous illustration. The frame flow from left to right is illustrated at the top of the illustration while the frame flow from right to left is shown at the bottom.

Figure 67 • **Up-VOE Location**



The illustration shows an Up-VOE, which resides on port 0, where the following applies:

- The Up-VOE Tx lookup is performed by the analyzer and it processes ingress OAM and data frames.
- The Up-VOE Rx lookup is performed by the rewriter and it processes egress OAM and data frames.

A single Up-VOE processes both egress (Rx) and ingress (Tx) frames.

3.23 VOP Common Functions

This section describes the functionality in the VOP that is common to all VOEs.

3.23.1 Accessing the VOP

For a given frame flow to be processed correctly by a VOE, the frame flow must be mapped to a VOE. Mapping the frame flow to a VOE includes configuring an ingress VOE lookup in the analyzer and an egress VOE lookup in the rewriter.

Frames are mapped to a service VOE the following way:

- In the analyzer, the service VOE lookup is configured via the IPT table using the frame ISDX as an index:
  ANA_CL:IPT:OAM_MEP_CFG.MEP_IDX_ENA = 1
  ANA_CL:IPT:OAM_MEP_CFG.MEP_IDX = [SERVICE VOE]
- In the rewriter, the service VOE lookup is configured through the ES0 action:
  ES0.OAM_MEP_IDX_VLD = 1
  ES0.OAM_MEP_IDX = [SERVICE VOE]

The frame flow must be mapped to the same VOE in the analyzer and the rewriter lookups.

Frames not mapped to a service VOE are processed by the port VOE. There is no way to prevent a lookup in the port VOE. However, if the port VOE is disabled, no processing is done.

3.23.2 VOE Hierarchy

For the purpose of supporting Loss Measurement of more than one hierarchical level, the device supports configuring a hierarchy of VOEs. A hierarchy of VOEs is required when OAM LM is run on more than one MEG level. In other words, when a service, which is being monitored by OAM LM, is transported across a server layer equally monitored by OAM LM. In that case, all the frames of service (including service OAM PDUs) must be counted as data frames by the LM counters at the server layer.

The LM hierarchy supported by the VOP depends on the type of VOE to which the frames are mapped. If the frame is mapped to a port VOE, only the LM counters of the port VOE are incremented, in this case there is no hierarchy. If the frame is mapped to a service VOE, the VOP supports a LM hierarchy of three. For example, a frame mapped to a service VOE is potentially counted by the LM counters at the following three layers:

- The frame is unconditionally counted at the service VOE Level.
- The frame may optionally be counted at a configurable server VOE layer.
- The frame is unconditionally counted in the port VOE.

The VOP supports this hierarchy in both the analyzer and the rewriter lookup.

3.23.2.1 Active VOE

Each frame is subject to one VOE lookup in the analyzer and one lookup in the rewriter. This implies that even when several levels of OAM LM counters need to be updated, only a single VOE processes the frame. As part of this processing, the VOE updates the LM counters of other VOEs in the VOE hierarchy. The VOE to which the frames are mapped is denoted the active VOE.

The active VOE can be either a port VOE or a service VOE. If the active VOE is a port VOE, only the LM counters of the port VOE are updated. If the active VOE is a service VOE, the active VOE always updates its own LM counters and the LM counters of the port VOE on the VOE resident port. It can optionally update LM counters of a configured server VOE.

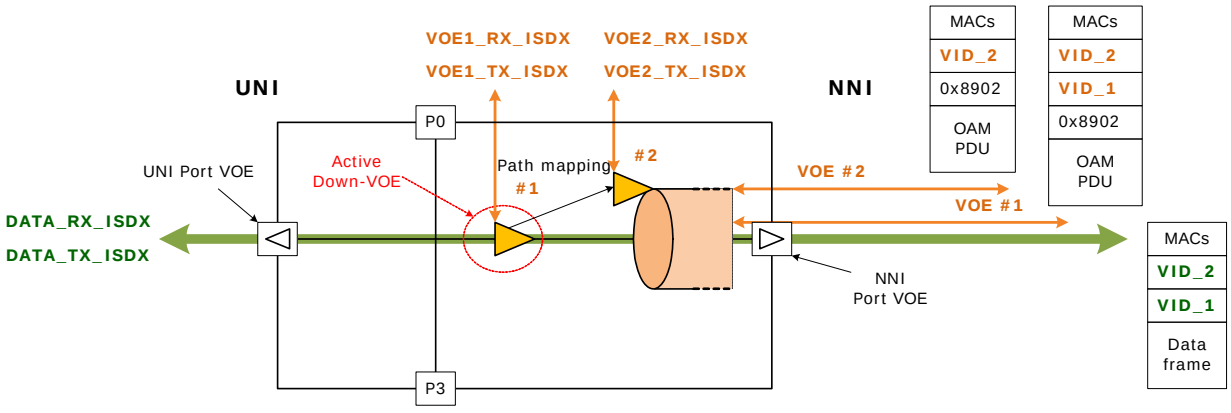
Considering a data frame that is forwarded from UNI to NNI; it can optionally be mapped to an Up-VOE in the analyzer (analyzer Tx lookup) and to a Down-VOE in the rewriter (rewriter Tx lookup). The active VOE in the analyzer Tx lookup is denoted the active Up-VOE, and the active VOE in the rewriter Tx lookup is denoted the active Down-VOE.

For a data frame in the opposite direction being forwarded from NNI to UNI, the active VOE in the analyzer Rx lookup is denoted the active Down-VOE, and the active VOE in the rewriter Rx lookup is denoted the active Up-VOE.

3.23.2.2 Down-VOE Hierarchy

The following illustration shows a Down-VOE hierarchy.

Figure 68 • **Down-MEP VOE Hierarchy**



A bidirectional data flow (indicated using green) is mapped to a service Down-VOE (Down-VOE #1). The service is carried across a server layer monitored by Down-VOE #2. Two OAM PDU flows are illustrated:

- Between Down-VOE #1 and its peer MEP (Service MEG)
- Between Down-VOE#2 and its peer MEP (Server MEG)

OAM PDU flows are indicated using orange. There can be additional OAM PDU flows between the NNI/UNI port VOEs and their peer MEPs. These flows are not shown in the figure.

Configure Down-VOE#2 as a Server VOE for Down-VOE#1:

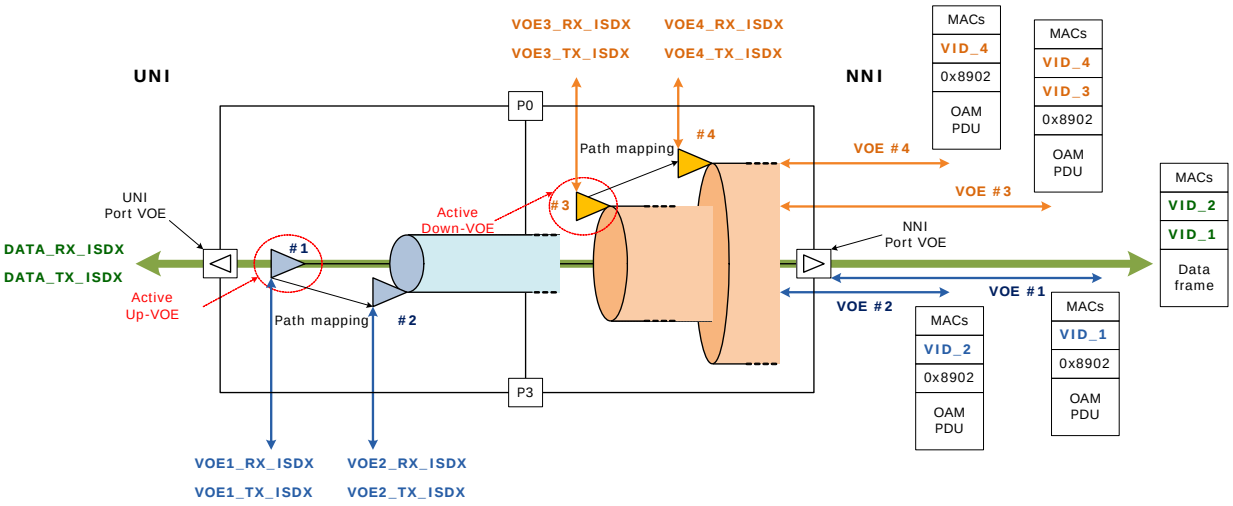
- Down-VOE#1 configuration
  VOP:VOE_CONF:PATH_VOE_CFG.PATH_VOE_ENA = 1
  VOP:VOE_CONF:PATH_VOE_CFG.PATH_VOEID = [VOE#2 IDX]
- Down-VOE#2 configuration
  VOP:VOE_CONF:VOE_CTRL.VOE_IS_PATH = 1

The Down-VOE#1 is the Active Down-VOE, so data frames and OAM PDUs to be processed by Down-VOE#1 must be mapped to Down-VOE#1 (analyzer and rewriter lookup). As part of the frame processing Down-VOE#1 updates the LM counters (Rx/Tx) of Down-VOE#2 and the LM counters of the NNI port VOE in addition to its own LM counters. OAM PDUs to be processed by Down-VOE#2 are mapped directly to Down-VOE#2. When processing this OAM, Down-VOE#2 is the Active Down-VOE, updating the LM counters of the NNI port VOE in addition to its own LM counters.

3.23.2.3 Up-VOE Hierarchy

The following illustration shows an Up-VOE hierarchy.

Figure 69 • **Up-MEP VOE hierarchy**



A bidirectional data flow (indicated using green) is mapped to a Service Up-VOE (VOE #1). The service is carried across a server layer monitored by Up-VOE #2. The device allows optionally mapping the data frames to two Down-VOEs as described previously. This is required when the up-server layer is carried across additional two layers of down-server layers. The mapping of the flow to two Down-VOEs and the NNI port VOE in addition to two Up-VOEs allows for creation of a VOE hierarchy of maximum five levels.

The following four OAM PDU flows are illustrated.

- The flow between VOE#1 and its peer MEP (Up-Service MEG)
- The flow between VOE#2 and its peer MEP (Up-Server MEG-1)
- The flow between VOE#3 and its peer MEP (Down-Server MEG-2)
- The flow between VOE#4 and its peer MEP (Down-Server MEG-3)

There can be additional OAM PDU flows between the NNI/UNI port VOEs and their peer MEPs. These flows are not shown in the illustration.

Configure Up-VOE#2 as a Server VOE for Up-VOE#1:

- Up-VOE#1 configuration
  VOP:VOE_CONF:PATH_VOE_CFG.PATH_VOE_ENA = 1
  VOP:VOE_CONF:PATH_VOE_CFG.PATH_VOEID = [Up-VOE#2 IDX]
- Up-VOE#2 configuration
  VOP:VOE_CONF:VOE_CTRL.VOE_IS_PATH = 1

For information about the required configuration of Down-VOE#3 and Down-VOE#4, see Down-VOE Hierarchy, page 226.

The Up-VOE#1 is the Active Up-VOE for data frames and OAM PDUs to be processed by Up-VOE#1; that is, these frames must be mapped to Up-VOE#1 (analyzer and rewriter lookup). As part of frame processing Up-VOE#1 updates the LM counters (Rx/Tx) of Up-VOE#2 and the UNI port VOE in addition to its own LM counters. OAM PDUs to be processed by Up-VOE#2 are mapped directly to Up-VOE#2. When processing this OAM, Up-VOE#2 is the Active Up-VOE, updating only its own LM counters. Because OAM PDUs processed in UP-VOE#2 will never pass the UNI port VOE, the UNI port VOE LM counters do not need to be updated.

3.23.3 VOE Frame Injection and Extraction

Tx OAM PDUs to be processed by a VOE must be injected into the VOE separately from frames received at a front port. Likewise Rx OAM PDUs which have been processed by a VOE can be extracted to a configurable CPU queue for further processing by a CPU.

Both Rx and Tx (injection) processing is modified if the VOE lookup is done with INDEPENDENT_MEL = 1, in which case, the OAM PDU is unconditionally processed as a data frame. In the following it is assumed that the VOE lookup is done with INDEPENDENT_MEL = 0. For more information about INDEPENDENT_MEL, see Independent MEL, page 239.

The VOE counts all frames discarded by the VOE. However, if an injected frame is discarded prior to reaching the VOE or if a frame extracted by the VOE is discarded prior to reaching the CPU, there are no counters to count these discards.

3.23.3.1 Frame Injection

Tx OAM PDUs must be injected into the VOE intended to process the PDU. For Ethernet VOEs it is important that the Ethernet OAM PDU is injected into the VOE at the correct pipeline point, because the VOE needs to detect whether the Tx PDU is to be processed (injected) or if the Tx PDU is to be subjected to Tx MEL filtering and subsequent forwarding. If the Tx PDU is not injected into the pipeline point of the Active VOE, it is submitted to MEL filtering, regardless whether it was received at a front port or injected at another place in the pipeline.

OAM PDUs injected into an Ethernet VOE must be injected with the correct MEL. If not the OAM PDU is either discarded (MEL_LOW) or processed as a data frame (MEL_HIGH). MPLS-TP VOEs process all injected Tx OAM PDUs unless INDEPENDENT_MEL = 1. When the Tx OAM PDU is correctly injected, the VOE automatically determines the Ethernet Opcode/MPLS-TP G-ACH Channel Type.

Tx OAM PDUs can be injected from the following three different sources.

- Internal CPU
- External CPU from a front port
- Automatic frame injection (AFI)

In all three cases, the injection is done using selected fields in the frame IFH, so Tx OAM PDUs are always injected using IFH. The following two tables show the IFH fields required to correctly inject a Tx OAM PDU to a Down-VOE and Up-VOE, respectively. Only IFH fields directly related to Tx injection are shown.

The fields listed in the following table are used when injecting Tx OAM PDUs into a Down-VOE.

Table 127 • **IFH Settings for Down-VOE Frame Injection**

IFH Field	Value	Description
MISC.PIPELINE_ACT	INJ	Pipeline command for Down-VOE injection
MISC.PIPELINE_PT	REW_IN_VOE REW_OU_VOE	Inject into Inner Down-VOE Inject into Outer Down-VOE
VSTAX.MISC.ISDX	[ISDX_TX]	OAM PDU Tx ISDX
VSTAX.MISC.COSID	[0-7]	Frame priority in Down-VOE
MISC.INJECT.DPORT	[NNI port]	Down-VOE residence port

Table 127 • **IFH Settings for Down-VOE Frame Injection *(continued)***

IFH Field	Value	Description
DST.ENCAP.PDU_TYPE	NONE OAM_Y1731 OAM_MPLS_TP SAM_SEQ	Data frame Y.1731 PDU MPLS-TP PDU SAM sequence numbering

The fields listed in the following table are used when injecting Tx OAM PDUs into an Up-VOE.

Table 128 • **IFH Settings for Up-VOE Frame Injection**

IFH Field	Value	Description
MISC.PIPELINE_ACT	INJ_MASQ	Pipeline command for Up-VOE injection
MISC.PIPELINE_PT	ANA_OU_VOE ANA_IN_VOE	Inject into Outer Up-VOE Inject into Inner Up-VOE
VSTAX.MISC.ISDX	[ISDX_TX]	OAM PDU Tx ISDX
VSTAX.MISC.COSID	[0-7]	Frame priority in Up-VOE
FWD.SRC_PORT	[UNI port]	Up-VOE residence port.
MISC.INJECT.DPORT	0	Must be set to zero.
DST.ENCAP.PDU_TYPE	NONE OAM_Y1731 OAM_MPLS_TP SAM_SEQ	Data frame Y.1731 PDU MPLS-TP PDU SAM sequence numbering

3.23.3.2 Frame Extraction

As part of the frame processing, the VOE can extract the frame to a configurable CPU queue based on the following criteria:

- Error condition (Rx/Tx)
- PDU type (Rx)

Only frames being processed as OAM PDU can be extracted, hence the VOE never extracts a frame being processed as a data frame. Based on the extract criteria, the VOE optionally extracts only the first frame matching the criteria (Hit-Me-Once) or all of the frames matching a given criterion. The destination of extracted frames is programmed into the following registers:

- Ethernet: VOP::CPU_EXTR_CFG.* and VOP::CPU_EXTR_CFG_1.*
- MPLS_TP: VOP::CPU_EXTR_MPLS.*

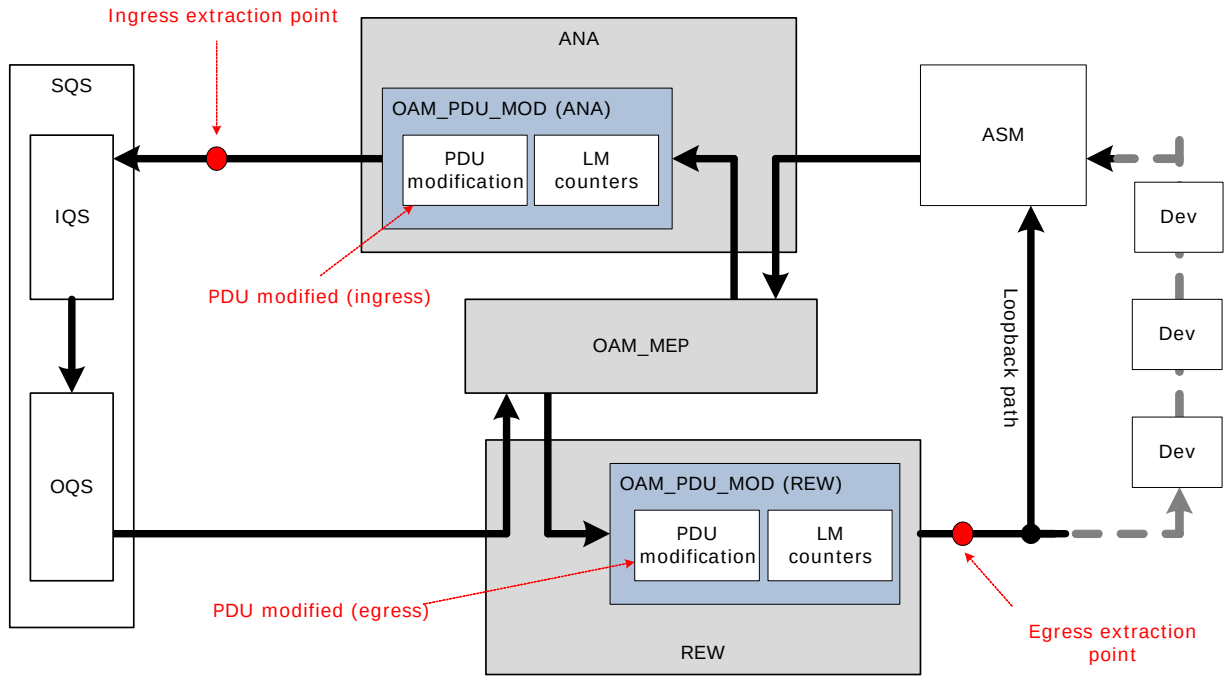
The above registers only allow extraction of frames to internal CPU queues. If it is required to extract frames to an external CPU on a front port, this must be done by forwarding relevant CPU queues to the front port.

In the Tx direction frames are only extracted due to error conditions, in which case frames are not modified. In the Rx direction, frames are extracted as part of normal operation for post processing in the CPU, so Rx OAM PDUs are modified by the VOE prior to extraction. The modifications done to the extracted Rx PDUs depend on the configuration of the VOE extracting the PDU.

The modification of Rx OAM PDUs depend on whether the VOE is configured for Down-VOE or Up-VOE and whether the extracted PDU is looped or not. OAM PDUs are modified by the OAM_PDU_MOD (analyzer) block the ingress direction and by the OAM_PDU_MOD (rewriter) in the egress direction. In addition to modifying the OAM PDU, frames being looped must have their MAC addresses swapped. MAC swapping is done in the rewriter. Modifying the frame encapsulation is done in the rewriter transparently to the VOP. For more information about rewriting the frame encapsulation, see Rewriter, page 322.

The following illustration shows the location of the OAM_PDU_MOD modules updating the OAM PDUs.

Figure 70 • **PDU Modification Location**



OAM PDUs extracted to the CPU at the ingress extraction point reflect the changes made to the PDU by the OAM_PDU_MOD (ANA). Likewise OAM PDUs extracted to the CPU at the egress extraction point reflect the changes made to the PDU by the OAM_PDU_MOD (rewriter).

A Down-VOE always extracts PDUs using the ingress extraction point. For an Up-VOE, the extraction point depends on whether the PDU is being looped or not. Looped frames are extracted in the ingress extraction point after being looped while non-looped frames are extracted in the egress extraction point.

The following are three important things to notice when extracting PDUs to the CPU:

- For PDU messages which can result in a PDU response the OpCode is modified at the egress modification point. This implies that when extracting a PDU Message at an Up-VOE, the OpCode of the extracted PDU depends on whether the PDU was looped or not. If the PDU is not looped, the OpCode is unaltered. If the PDU is looped, the extracted PDU contains the updated OpCode. This affect the following PDU types LMM, DMM, LBM, and SLM. For a Down-VOE the extracted PDU always matches the OpCode of the received PDU.
- If a PDU is looped in an Up-VOE, the extraction to the CPU is done in the ingress extraction point. This implies that the copy being sent to the CPU is actually a copy of the Response being transmitted to the Peer MEP, rather than a copy of the CPU received from the remote MEP. Hence it reflects any changes made in the OAM_PDU_MOD (ANA) due to the Tx lookup.
- Swapping of MAC addresses for Message to Response loopback is only done if the PDU is looped. Hence if the Message is extracted to the CPU without looping the MAC addresses are not swapped.

The modifications done to the PDU message are PDU specific.

3.23.4 Loss Measurement (LM) Counters

The Ethernet VOE functions include two kinds of loss measurements. They both use the same LM counters. This section describes how the LM counters are accessed by the CPU and updated by the VOE. LM counters are only valid when the VOE is configured for Ethernet. If the VOE is configured for MPLS, no LM counters are supported.

The VOP supports two kinds of Y.1731 Ethernet loss measurements (LM):

- Frame loss measurement (F-LM)
  - Single-ended LM (LMM/LMR)
  - Dual-ended LM (CCM-LM)

- Synthetic loss measurement (SynLM)
  - Single-ended SynLM (SLM/SLR)
  - Dual-ended SynLM (1SL)

Ethernet VOEs can be configured to support one of the LM flavors.

3.23.4.1 VOE LM Counters

Each VOE has eight 32-bit LM counters in OAM_PDU_MOD (ANA) and eight 32-bit LM counters in OAM_PDU_MOD (rewriter). Down-VOE LM counters are accessed as follows:

- Service VOE Tx counters (REW:VOE_SRV_LM_CNT:SRV_LM_CNT_LSB)
- Port VOE Tx counters (REW:VOE_PORT_LM_CNT.PORT_LM_CNT_LSB)
- Service VOE Rx counters (ANA_AC_OAM_MOD:VOE_SRV_LM_CNT.SRV_LM_CNT_LSB)
- Port VOE Rx counters (ANA_AC_OAM_MOD:VOE_PORT_LM_CNT.PORT_LM_CNT_LSB)

Up-VOE LM counters are accessed as follows:

- Service VOE Tx counters (ANA_AC_OAM_MOD:VOE_SRV_LM_CNT.SRV_LM_CNT_LSB)
- Service VOE Tx counters (REW:VOE_SRV_LM_CNT:SRV_LM_CNT_LSB)

3.23.4.2 Frame Loss Measurement (F-LM)

An Ethernet VOE is configured for F-LM VOP:VOE_CONF_REG:VOE_MISC_CONFIG.SL_ENA = 0

When configured for F-LM, the LM counters are updated based on frame priority, supporting eight priorities. That is, the VOE counts the number of frames of a given priority that have entered (Tx) or left (Rx) the connection being monitored by the VOE. The F-LM frame priority is represented by IFH.COSID.

The VOE includes all data frames in the F-LM count. In addition, all PDUs being processed by the VOE can optionally be included in the F-LM count. This is configurable per PDU per VOE. OAM PDUs that fail the OAM validation are not included in the F-LM count.

3.23.4.3 F-LM COSID Mapping

To support F-LM in a VOE hierarchy, the VOP supports COSID mapping. This is required because a frame can have different priorities at different levels in a VOE hierarchy. The mapping affects both the frame priority and the frame color. For more information about VOE hierarchy, see VOE Hierarchy, page 225.

The mapping differs between service VOEs and port VOEs and it differs depending on whether a VOE processes the OAM PDU or whether the OAM PDU is passed transparently, for example, as a data frame or due to MEL high.

A pre-condition for the VOE COSID mapping to work is that the classifier classifies the frame IFH.COSID as follows:

- OAM PDU: IFH.COSID is set to the priority given by the VOE processing the frame. For example. if an OAM PDU is processed by the Outer Up-VOE, the IFH.COSID is set to the priority dictated by the outer Up-VOE.
- Data frame/MEL high: If the IFH.COSID is used for mapping the frame priority, the value must be set to the priority given by the service VOE closest to the UNI to which the frame is mapped. This is most often the VOE implementing the EVC MEP.

If the frame is not mapped to a service VOE, the IFH.COSID is not used for updating the F-LM counters.

The IFH.COSID is set to the frame priority given by the VOE processing the OAM PDU, hence a VOE always increments its own F-LM counter based on the IFH.COSID when processing an OAM PDU, without any mapping. Frame color is given unconditionally by IFH.DP.

It is configurable per service VOE if the F-LM counts only green frames or both green and yellow frames:

- VOP:ANA_COSID_MAP_CONF:COSID_MAP_CFG_ANA.CNT_YELLOW_ANA
- VOP:REW_COSID_MAP_CONF:COSID_MAP_CFG_REW.CNT_YELLOW_REW

This is configured separately in the analyzer and in the rewriter, but for all normal scenarios, they must be set to the same value for any given VOE.

When a frame is mapped to a service VOE but it is not processed as an OAM PDU, the source of the frame COSID and color is configurable and the COSID is mapped. This includes data frames, OAM PDUs with MEL HIGH, or OAM PDUs which belong to another MEG domain than the current VOE.

The COSID and color mapping consists of two steps:

- Select the source of the mapping (COSID/Color)
  Configuring the source of the VOE COSID mapping:
  VOP:ANA_COSID_MAP_CONF:COSID_MAP_CFG_ANA.COSID_SRC_SEL_ANA
  VOP:REW_COSID_MAP_CONF:COSID_MAP_CFG_REW.COSID_SRC_SEL_REW

  The COSID source is set separately in the analyzer and the rewriter, but for all normal scenarios, they must be set to the same value for any given VOE.

  Configuring the source of the VOE Color:
  VOP:ANA_COSID_MAP_CONF:COSID_MAP_CFG_ANA.COLOR_SRC_SEL_ANA
  VOP:REW_COSID_MAP_CONF:COSID_MAP_CFG_REW.COLOR_SRC_SEL_REW

  The color source is set separately in the analyzer and the rewriter, but for all normal scenarios, they must be set to the same value for any given VOE.
- Select the mapping table (COSID):
  VOP:ANA_COSID_MAP_CONF:COSID_MAP_TABLE_ANA.COSID_MAP_TABLE_ANA
  VOP:REW_COSID_MAP_CONF:COSID_MAP_TABLE_REW.COSID_MAP_TABLE_REW

  The mapping table is configured separately in the analyzer and the rewriter, but for all normal scenarios, they must be configured identically.

Port VOEs perform F-LM COSID mapping differently than service VOEs. There is no difference between how OAM PDUs processed by the port VOE and other frames are counted. The basis of the COSID mapping and the frame color is always based on the outer VLAN [PCP, DEI] bits alternatively port default values. For Rx frames [PCP, DEI] is extracted from the ingress frames received on the front port. For Tx frames [PCP, DEI] is extracted from the egress frames after rewriting.

The port VOE [PCP,DEI] mapping is configured as follows:

- VOP:PORT_COSID_MAP_CONF:PORT_RX_COSID_MAP.PORT_RX_COSID_MAP
- VOP:PORT_COSID_MAP_CONF:PORT_RX_COSID_MAP1.PORT_RX_COSID_MAP1
- VOP:PORT_COSID_MAP_CONF:PORT_TX_COSID_MAP.PORT_TX_COSID_MAP
- VOP:PORT_COSID_MAP_CONF:PORT_TX_COSID_MAP1.PORT_TX_COSID_MAP1

There is no explicit configuration of whether to count yellow frames, this is embedded in the COSID mapping above.

3.23.4.4 Synthetic Loss Measurement (SynLM)

An Ethernet VOE is configured for SynLM in VOP:VOE_CONF_REG:VOE_MISC_CONFIG.SL_ENA = 1.

When configured for SynLM, the LM counters are used to count only valid SLM/SLR/1SL PDUs being processed at the VOE; no other frames are counted. For example, if a port VOE is configured for SynLM, the port VOE LM counters do not include any other frames passing the port even when mapped to a service VOE located on that port.

The frame priority is no longer used to determine the index of which LM counter to update. Instead the counter index is determined by the SynLM remote Peer IDX, without any kind of mapping. For more information about implementing SynLM, see Synthetic Loss Measurement, page 254.

The configuration of whether to count yellow frames for service and port VOEs, respectively, is the same as for F-LM described previously.

3.23.5 Port Count-All Rx/Tx Counters

For every physical port, the VOP counts the number of bytes and frames that pass the location of the port VOE in the Rx/Tx direction. Frames (32-bit counter) and bytes (40-bit counter) are counted per COSID. The COSID is determined by the configuration of the port MEP located on each port. Because the

counters are located in the OAM_PDU_MOD, the Rx counters are located in the ANA, and the Tx counters are located in the REW.

Whenever one of the port count-all counters (or the port VOE LM counter) is read, all of the port count-all counters (and the port VOE LM counter) located in the same OAM_PDU_MOD and with the same COSID are sampled into a register for subsequent reading. This allows the CPU to sample all the counters at the same time.

The port count-all Tx counters:

- REW:VOE_PORT_LM_CNT:PORT_FRM_CNT_LSB.PORT_FRM_CNT_LSB
- REW:VOE_PORT_LM_CNT:PORT_BYTE_CNT_MSB.PORT_BYTE_CNT_MSB
- REW:VOE_PORT_LM_CNT:PORT_BYTE_CNT_LSB.PORT_BYTE_CNT_LSB

The port count-all Tx counters are sampled into the following sample registers:

- REW::RD_LAST_PORT_LM_CNT_LSB.RD_LAST_PORT_LM_CNT_LSB
- REW::RD_LAST_PORT_FRM_CNT_LSB.RD_LAST_PORT_FRM_CNT_LSB
- REW::RD_LAST_PORT_BYTE_CNT_MSB.RD_LAST_PORT_BYTE_CNT_MSB
- REW::RD_LAST_PORT_BYTE_CNT_LSB.RD_LAST_PORT_BYTE_CNT_LSB

The port count-all Rx counters:

- ANA_AC_OAM_MOD:VOE_PORT_LM_CNT:PORT_FRM_CNT_LSB.PORT_FRM_CNT_LSB
- ANA_AC_OAM_MOD:VOE_PORT_LM_CNT:PORT_BYTE_CNT_MSB.PORT_BYTE_CNT_MSB
- ANA_AC_OAM_MOD:VOE_PORT_LM_CNT:PORT_BYTE_CNT_LSB.PORT_BYTE_CNT_LSB

The port count-all Rx counters are sampled into the following sample registers:

- ANA_AC_OAM_MOD::RD_LAST_PORT_LM_CNT_LSB.RD_LAST_PORT_LM_CNT_LSB
- ANA_AC_OAM_MOD::RD_LAST_PORT_FRM_CNT_LSB.RD_LAST_PORT_FRM_CNT_LSB
- ANA_AC_OAM_MOD::RD_LAST_PORT_BYTE_CNT_MSB.RD_LAST_PORT_BYTE_CNT_MSB
- ANA_AC_OAM_MOD::RD_LAST_PORT_BYTE_CNT_LSB.RD_LAST_PORT_BYTE_CNT_LSB

3.23.6 Basic VOP Configuration

The VOP is enabled in VOP::VOP_CTRL.VOP_ENA. When not enabled, the VOP can be configured, but no frames are processed.

3.23.7 Loss of Continuity Controller

The VOP contains a loss of continuity controller (LOCC). LOCC is used for the following:

- Detecting loss of continuity
  - Ethernet, CCM(-LM) frames are not received from peer MEP
  - MPLS-TP, BFD-CC frames are not received from peer MEP.
- CCM-LM auto insertion. Ethernet: Insert LM information into Tx CCM PDUs at a programmable interval.

The LOCC implements seven LOC timers that can be programmed to expire at different intervals. For an Ethernet VOE, the LOC counters are used for the following:

- LOC detection. Each VOE is optionally assigned an LOC timer for LOC detection (VOP:VOE_CONF:CCM_CFG.CCM_PERIOD). When the assigned LOC timer expires, the VOE increments the CCM miss counter, which is cleared when a valid CCM(-LM) PDU is received. If the CCM miss counter reaches a fixed value, the VOE generates an LOC event.
- CCM-LM auto insertion. Each VOE is optionally assigned an LOC timer for CCM-LM insertion (VOP:VOE_CONF:CCM_CFG.CCM_LM_PERIOD). When the assigned LOC timer expires, the next CCM injected into the VOE (Tx) is processed as CCM-LM.

For an MPLS-TP VOE, the LOC counters are used for LOC detection. Each VOE is optionally assigned an LOC timer for LOC detection (VOP_MPLS:VOE_CONF_MPLS:BFD_CONFIG.BFD_SCAN_PERIOD).

When the assigned LOC timer expires, the VOE increments the BFD miss counter, which is cleared when a valid BFD PDU is received. If the CCM miss counter reaches a programmable value, the VOE generates an LOC event.

The HMOC is based on the same logic as the LOCC. The two HMO timers are implemented as an extension to the seven LOC timers in all bit masks related to the LOCC.

3.23.7.1 LOC Timer Configuration

Each time an LOC timer expires, the LOCC scans through all the VOEs and updates LOC miss counters in VOE to which the expired LOC timer is assigned. The process of scanning all the VOEs to update based on LOC timer expiry is called an LOC scan.

The LOC scan scans a new VOE every clock cycle in which the VOP is not processing a frame or a accessed by a CSR access. This implies that the minimum time required to complete an LOC scan is 1.299 µs (203 VOEs × 6.4 ns/VOE). However, a scan is only achieved if there are no VOE lookups and no register accesses to the VOP. Each frame access and each register access delays the LOC scan by a single system clock cycle.

To start the LOC timers, the LOC scan function must be enabled (VOP::VOP_CTRL.LOC_SCAN_ENA). If the LOC SCAN is not enabled, LOC timers are not incremented and they never expire.

The LOCC implements a counter that generates a base tick every time a configurable number of system clock cycles have passed. The period of the common LOC base tick is configured in VOP::LOC_CTRL.LOC_BASE_TICK_CNT. The LOC_BASE_TICK_CNT is configured as the number of system clock cycles between two adjacent LOC base ticks.

Each LOC timer is incremented for every base tick and expires after an individually configured number of base ticks. The LOC timer expiration period is configured in VOP::LOC_PERIOD_CFG.LOC_PERIOD_VAL.

Y.1731 requires the LOC condition to be signaled after 3.5 times the CCM period. To implement this, the LOC timers expire after half the configured period, scanning the VOE array for every half CCM period. The Ethernet VOEs assert the LOC signals if seven consecutive scans occur without receiving a valid CCM(-LM) frame.

3.23.7.2 LOC Scan Control Signals

To force a LOC scan of selected LOC timers without any LOC timer expiry, enable VOP::LOC_CTRL.LOC_FORCE_HW_SCAN_ENA[6:0]. An LOC scan is performed for every LOC timer indicated by a value of 1 in the bit mask written to the this register.

To force an HMO scan, enable VOP::LOC_CTRL.LOC_FORCE_HW_SCAN_ENA[7:8].

To stop an ongoing scan, disable the LOC SCAN controller by setting VOP::VOP_CTRL.LOC_SCAN_ENA = 0. This causes the LOC scan controller to reset (all LOC timers reset to zero). Note that this does not imply that the individual VOE LOC miss counters are cleared.

3.23.7.3 LOC Scan Status Bits

The current state of the LOC scan is captured in VOP::LOC_SCAN_STICKY.LOC_SCAN_ONGOING_STATUS. The register returns a bit for each of the seven LOC timers (bits [6:0] and two HMO timers (bits [7:8]). The bit corresponding to a given LOC timer is asserted in this mask, if the currently active LOC scan includes a timeout of the LOC counter. So, if bit 1 is asserted in the returned mask, the current scan is updating all VOEs to which LOC timer 1 is assigned.

The VOP::LOC_SCAN_STICKY.LOC_SCAN_COMPLETED_STICKY sticky bit is asserted every time an LOC scan is completed. The VOP::LOC_SCAN_STICKY.LOC_SCAN_STARTED_STICKY sticky bit is asserted every time an LOC scan is initiated.

If a given LOC timer expires before the previous LOC scan initiated by this LOC timer was completed, the VOE asserts the VOP::LOC_SCAN_STICKY.LOC_SCAN_START_DELAYED_STICKY sticky bit. This is an error condition, and indicates that the LOC timer has a timeout period, which is smaller than the lowest LOC scan time supported by the VOP.

3.23.8 Hit-Me-Once Controller

The VOP contains a Hit-Me-Once controller (HMOC). HMOC is used to assert the VOE extraction bits for selected VOEs at programmable intervals to allow periodic extraction of selected PDU types to the CPU

without CPU intervention. The process of automatically updating the VOE extraction bits is referred to as “auto HMO”.

The HMOC implements two HMO timers that can be programmed to expire at different intervals (for example, one fast timer and one slow timer). The HMO timers are implemented in the same way as the LOCC. The HMOC and the LOCC use common logic when scanning through the VOE array to update the LOC and auto HMO registers respectively.

The LOCC control signals also apply to the HMOC.

3.23.8.1 HMO Timer Configuration

When an HMO timer expires, the HMOC scans through all the VOEs and updates the VOE extraction bits based on VOP and VOE configuration. Scanning the VOEs to update VOE extraction bits, is known as an HMO scan. HMO scanning is done using the same logic as an LOC scan. The two HMO timers can be seen as an addition to the seven LOC timers.

To start the HMO timers, the LOC scan function must be enabled (VOP::VOP_CTRL.LOC_SCAN_ENA). If the LOC SCAN is not enabled, HMO timers are not incremented, and they never expire.

Each HMO timer is incremented for every LOCC base tick and expires after an individually configured number of base ticks. The HMO timer expiration period is configured in VOP::HMO_PERIOD_CFG.HMO_PERIOD_VAL.

The expiration period for the HMO timers is programmed in the same way as for LOC timers. For more information, see LOC Timer Configuration, page 234.

In addition to the HMO expiry counter, each HMO timer maintains a 3-bit wrapping HMO slot counter, which is incremented every time the HMO timer expires. The HMO slot counter cannot be read by the CPU. When an HMO timer expires, an HMO scan is initiated with the current value of the HMO slot counter. This value is referred to as the active HMO slot for this timer.

When a VOE is configured for auto HMO, it is configured with an HMO slot value. During an HMO scan, the active HMO slot is compared to an HMO slot configured for every VOE. Only VOEs configured with an HMO slot value equal to the active HMO slot are updated.

For example, every VOE is updated every 8th time the HMO timer expires. This spaces the frames extracted from the VOEs as a result of auto HMO to avoid flooding the CPU when an HMO timer expires. This also means that the HMO expiry timer must be programmed to be eight times faster than the desired extraction period, because it must expire eight times for all the VOEs to be updated.

3.23.8.2 HMO Scan Control Signals

An HMO scan of selected HMO timers without any HMO timer expiry can be forced in VOP::LOC_CTRL.LOC_FORCE_HW_SCAN_ENA[7:8]. An HMO scan is performed for every HMO timer indicated by a value of 1 in the bit mask written to this register.

When an HMO scan is forced, the HMO slot value to be used in the scan is configured in the VOP::HMO_FORCE_SLOT_CFG.HMO_FORCE_SLOT register.

3.23.8.3 HMO Timer Assignment

The auto HMO function can assert the VOE extraction bit of selected PDU types. The PDU types that can be extracted by the auto HMO function are divided into five groups. The VOE extraction bits belonging to each HMO group are listed.

- Valid CCM frame (HMO group 1), VOP:VOE_STAT:PDU_EXTRACT.CCM_RX_CCM_NXT_EXTR
- Valid CCM frame with non-zero end TLV (HMO group 2), VOP:VOE_STAT:PDU_EXTRACT.CCM_RX_TLV_NON_ZERO_EXTR
- Invalid CCM frame (HMO group 3)
  - VOP:VOE_STAT:PDU_EXTRACT.CCM_ZERO_PERIOD_RX_ERR_EXTR
  - VOP:VOE_STAT:PDU_EXTRACT.RX_MEL_LOW_ERR_EXTR
  - VOP:VOE_STAT:PDU_EXTRACT.CCM_MEGID_RX_ERR_EXTR
  - VOP:VOE_STAT:PDU_EXTRACT.CCM_MEPID_RX_ERR_EXTR
  - VOP:VOE_STAT:PDU_EXTRACT.CCM_PERIOD_RX_ERR_EXTR
  - VOP:VOE_STAT:PDU_EXTRACT.CCM_PRIO_RX_ERR_EXTR

- Test frame (TST/ LBR/NON_OAM_MSG) (HMO group 4), VOP:VOE_STAT:PDU_EXTRACT.RX_TEST_FRM_NXT_EXTR
- Valid SLR frame (HMO group 5), VOP:VOE_STAT:SYNLM_EXTRACT.EXTRACT_PEER_RX.

When auto HMO is enabled for the HMO groups, the extraction will always be done Hit-Me-Once, regardless of the HMO configuration in VOP:VOE_STAT:PDU_EXTRACT.EXTRACT_HIT_ME_ONCE.

Each of these groups is assigned one of the HMO timers using the following VOP configuration:

- VOP::HMO_TIMER_CFG.HMO_RX_CCM_NXT_TIMER (HMO group 1)
- VOP::HMO_TIMER_CFG.HMO_CCM_RX_TLV_NON_ZERO_TIMER (HMO group 2)
- VOP::HMO_TIMER_CFG.HMO_CCM_RX_BAD_NXT_TIMER (HMO group 3)
- VOP::HMO_TIMER_CFG.HMO_RX_TEST_FRM_NXT_TIMER (HMO group 4)
- VOP::HMO_TIMER_CFG.HMO_EXTRACT_PEER_RX_TIMER (HMO group 5)

In other words, at the VOP level, two HMO timers can be configured (one fast and one slow). Each of the auto HMO groups are then assigned to one of the HMO timers.

3.23.8.4 VOE HMO Configuration

Every VOE is assigned an auto HMO slot (VOP:VOE_STAT:AUTO_HIT_ME_ONCE.HMO_SLOT).

In a VOE, auto HMO is enabled individually for every HMO group

- HMO group 1: VOP:VOE_STAT:AUTO_HIT_ME_ONCE.HMO_CCM_RX_CCM_NXT
- HMO group 2: VOP:VOE_STAT:AUTO_HIT_ME_ONCE.HMO_CCM_RX_TLV_NON_ZERO
- HMO group 3: VOP:VOE_STAT:AUTO_HIT_ME_ONCE.HMO_CCM_RX_BAD_NXT
- HMO group 4: VOP:VOE_STAT:AUTO_HIT_ME_ONCE.HMO_RX_TEST_FRM_NXT
- HMO group 5: VOP:VOE_STAT:AUTO_HIT_ME_ONCE.HMO_EXTRACT_PEER_RX

When an HMO timer expires, an HMO scan is initiated for the active HMO timer and the active HMO slot. During an HMO scan the HMO group of an individual VOE is updated if:

- The active HMO slot equals the auto HMO slot configured in the VOE.
- The HMO timer assigned to the HMO group is active in the current HMO scan.
- The HMO group is enabled for auto HMO in the VOE.

For example, when HMO timer 0 expires with active HMO slot = 5, an HMO scan is started. The scan will traverse all the VOEs and update all the VOEs that are configured with an HMO slot count of five.

For all VOEs that are configured with auto HMO slot = 5, the auto HMO groups that are assigned to HMO timer 0 are updated. For more information, see HMO Timer Assignment, page 235.

For groups 1, 2, 3, and 4, an “auto HMO update” implies that the corresponding VOE extraction bits are asserted if the group is enabled for auto HMO in the VOE.

For group 5, an “auto HMO updated” implies that the value of the auto HMO enable register is written to the corresponding VOE extraction register.

In other words, the value of the enable register VOP:VOE_STAT:AUTO_HIT_ME_ONCE.HMO_EXTRACT_PEER_RX is written to the VOE extraction register, VOP::HMO_TIMER_CFG.HMO_EXTRACT_PEER_RX_TIMER.

3.23.9 Interrupt Controller

The VOP implements a single interrupt connected to the interrupt controller in the VCore labeled OAM_VOP. For more information, see Interrupt Controller, page 221.

The VOP interrupt is enabled in VOP::MASTER_INTR_CTRL.OAM_MEP_INTR_ENA. The current VOP interrupt status is available in VOP::MASTER_INTR_CTRL.OAM_MEP_INTR. The VOP interrupt is asserted if one or more of the VOEs assert their individual interrupt.

The status of the individual VOE interrupts is available in VOP::INTR.VOE_INTR.

To quickly determine which VOEs have active interrupts, the VOP implements a register in which each bit is a “logical or” of each of the 32-bit registers found in VOP::INTR.VOE_INTR. In other words, if any of the 32 interrupts in VOP::INTR.VOE_INTR[x] is asserted, the corresponding bit *n* is asserted in the VOP::VOE32_INTR.VOE32_INTR register.

An example of the mapping between VOE32_INTR and VOE_INTR is as follows:

- VOP::VOE32_INTR.VOE32_INTR bit 0 → VOP::INTR.VOE_INTR[0] (32 bits)
- VOP::VOE32_INTR.VOE32 bit 1 → VOP::INTR.VOE_INTR[1] (32 bits)

The interrupt controller is common to all Ethernet VOEs and MPLS-TP VOEs. For more information about the interrupt sources, see VOE: Ethernet OAM, page 239 for Ethernet OAM and VOE: MPLS-TP OAM, page 274 for MPLS-TP OAM.

3.23.10 Ethernet Configuration

The section describes the VOP configuration common to all Ethernet VOEs.

3.23.10.1 G.8113.1 Support

It must be configured in VOP:VOE_CONF:VOE_CTRL.G_8113_1_ENA if the Ethernet VOE is to process Y.1731 PDUs or 6.8113.1 PDUs.

3.23.10.2 CPU Extraction Queues

The VOE optionally extracts selected frames based on different criteria. The destination of the extracted frames is configured at VOP level in VOP::CPU_EXTR_CFG and VOP::CPU_EXTR_CFG_1. The destinations are internal CPU queues. To extract frames to a front port, the internal CPU queues must be forwarded to the required front port.

All OAM PDUs extracted to due to VOE Rx or Tx validation errors are extracted to CPU error queue VOP::CPU_EXTR_CFG.CPU_ERR_QU.

For more information about extracting frames to the CPU, see VOE: Ethernet OAM, page 239.

3.23.10.3 PDU Versions

As part of the PDU Rx validation the version of each PDU can be verified against programmable values. The valid PDU versions are configured in VOP::VERSION_CTRL and VOP::VERSION_CTRL_2.

3.23.10.4 VOE Multicast Address

As part of the PDU Rx validation the frame DMAC can be verified against a single programmable multicast MAC address common to all VOEs. The VOP multicast address is configured in VOP::COMMON_MEP_MC_MAC_MSB.MEP_MC_MAC_MSB and VOP::COMMON_MEP_MC_MAC_LSB.MEP_MC_MAC_LSB.

3.23.10.5 Generic PDU Support

The Y.1731 PDUs have dedicated hardware support in the VOE. For more information, see Ethernet OAM (Y.1731 and G.8113.1 OAM), page 222. In addition to these PDUs, the VOE supports eight configurable Generic OpCodes. A generic OpCode is an Y.1731 OpCode that does not have dedicated VOE support. The eight Generic OpCodes are configured per VOP, and the Generic OpCode processing is configured separately for each VOE. A VOE supports the following for Generic OpCodes:

- Extraction to a dedicated CPU queue.
- Simple statistics: Count in selected or nonselected OAM counter.
- Configuration of whether frames with Generic OpCodes are counted as data in the VOE LM counters.
- Sticky bits to indicate reception of a valid Rx generic OpCode.

The VOP supports configuration of eight Generic OpCodes in VOP::OAM_GENERIC_CFG.GENERIC_OPCODE_VAL, VOP::OAM_GENERIC_CFG.GENERIC_OPCODE_CPU_QU, and VOP::OAM_GENERIC_CFG.GENERIC_DMAC_CHK_DIS.

When an Y.1731 PDU is mapped to the VOE, the VOE determines if the OpCode is configured as a Generic OpCode. If this is the case, the Rx PDU is processed as a Generic OpCode. The default processing of hardware-supported PDUs can be overridden by configuring any of the hardware-supported OpCodes as a Generic OpCode.

If the PDU OpCode is not recognized as a Generic OpCode or a known OpCode, the PDU is classified as an Unknown OpCode. Unknown OpCodes can be extracted to the CPU Default Queue.

Rx PDUs are validated before being classified as either a valid Generic OpCode or as a valid Unknown Opcode. For more information about validating Rx PDUs, see OAM PDU Validation: Rx Direction, page 240.

3.23.10.6 CCM-LM Rx: Update Reserved Field

The CCM-LM PDU specified in Y.1731 does not contain a field for carrying the RxFCf counter value. The VOE can write the RxFCf value into the CCM.Reserved field when receiving a valid CCM-LM PDU. This is the only way to convey the RxFCf value to the CPU.

Writing the RxFCf value into the CCM_LM.Reserved field is enabled in VOP::VOP_CTRL.CCM_LM_UPD_RSV_ENA.

3.23.10.7 LMR Rx: Overwrite End TLV

The LMR PDU specified in Y.1731 does not contain a field for carrying the RxFCb counter value. The VOE can write the RxFCb value into the 4 bytes following the LMR.TxFCb field when receiving a valid LMR PDU, hereby overwriting the LMR.End_TLV field. This is the only way to convey the RxFCb value to the CPU.

In addition to inserting the RxFCb value, the VOE sets the LMR.TLV field = 16 and sets the byte following the newly inserted RxFCb field = 0, hereby creating a new LMR.End_TLV.

Writing the RxFCb value into the LMR PDU is enabled in VOP::VOP_CTRL.LMR_UPD_RXFCL_ENA.

3.23.10.8 Detect CCM(-LM) Port Change Count

The VOE detects when the source port of the Rx CCM(-LM) PDUs changes due to for instance ring protection failover. After a source port change, the VOE counts the number of consecutive CCM-LM PDUs from a given port. When this count reaches a programmable value, the source port is considered stable. The number of consecutive CCM(-LM) PDUs required to consider the port stable is configured in VOP::VOP_CTRL.CCM_RX_SRC_PORT_DETECT_CNT.

3.23.11 MPLS-TP Configuration

The section describes the VOP configuration common to all MPLS-TP VOEs.

3.23.11.1 CPU Extraction Queues

The VOE can extract selected frames based on different criteria. The destination of the extracted frames is configured at VOP level in VOP::CPU_EXTR_MPLS.

The destinations are internal CPU queues. To extract frames to a front port, the internal CPU queues must be forwarded to the required front port. The CPU error queue for MPLS-TP frames is the same queue as Ethernet VOEs (VOP::CPU_EXTR_CFG.CPU_ERR_QU).

For more information about extracting frames to the CPU, see VOE: MPLS-TP OAM, page 274.

3.23.11.2 BFD Version

As part of the PDU Rx validation, the version of each PDU can be verified against a programmable value in VOP::VERSION_CTRL_MPLS.BFD_VERSION. For more information about MPLS-TP PDU Rx validation implemented by the VOE, see BFD Rx Validation, page 278.

3.23.11.3 Generic G-ACH Channel Types

The VOE supports eight configurable Generic G-ACH Channel Types. For more information about the MPLS-TP PDUs that have dedicated hardware support in the VOE, see MPLS-TP OAM (G.8113.2 OAM), page 222.

A Generic G-ACH Channel Type is an MPLS-TP Channel Type that does not have dedicated VOE support. The eight Generic G-ACH Channel Types are configured per VOP, and the Generic G-ACH Channel Type processing is configured separately for each VOE.

VOE supports the following for Generic G-ACH Channel Types:

- Extraction to a dedicated CPU queue.
- Simple statistics: Count in selected/nonselected OAM counter.

- Sticky bits to indicate reception of a valid Rx of a Generic G-ACH Channel Type.

The VOP supports configuration of eight Generic G-ACH Channel Types in VOP::MPLS_GENERIC_CODEPOINT.GENERIC_CODEPOINT_VAL and VOP::MPLS_GENERIC_CODEPOINT.GENERIC_CODEPOINT_CPU_QU.

When a MPLS-TP PDU is mapped to a VOE, the VOE determines if the PDU G-ACH Channel Type is configured as a Generic G-ACH Channel Type. If this is the case, the Rx PDU is processed as a Generic G-ACH Channel Type. The default processing of hardware-supported PDUs can be overridden by configuring any of the hardware-supported G-ACH Channel Types as a Generic G-ACH Channel Type.

If the PDU G-ACH Channel Type is not recognized as a Generic Channel Type or as a known G-ACH Channel Type the PDU is classified as an Unknown G-ACH Channel Type. Rx PDUs with unknown G-ACH Channel Type can be extracted to the CPU default queue.

3.24 VOE: Ethernet OAM

Within the scope of VOE configuration and processing, the term “Ethernet OAM” denotes OAM as specified by Y.1731, and Y.1731-like OAM as specified by G.8113.1 (MPLS-TP OAM). For processing of both Y.1731 and G.8113.1 OAM, the VOE must be configured as an Ethernet VOE. Whether the VOE must process Y.1731 or G.8113.1 OAM is determined by the following configuration in VOP:VOE_CONF:VOE_CTRL.G_8113_1_ENA.

The following description of Ethernet VOE functions applies to both Y.1731 and G.8113.1 OAM, unless specifically noted. For information about G.8113.1 specific functions, see G.8113.1 Specific Functions, page 272.

3.24.1 Ethernet VOE Functions

This section describes the Ethernet VOE functionality that is common to all Ethernet PDUs. The next sections describe the Ethernet VOE processing of supported PDUs.

The first step of the Rx and Tx frame processing in the VOE is to validate the frame to determine if it is a valid frame to be processed or an invalid frame to be discarded.

3.24.1.1 Independent MEL

The VOE validates and processes OAM PDUs while data frames are not validated and never processed. In some scenarios, it is required to map valid OAM PDUs to a VOE but to process these as data frames; that is, bypass the OAM PDU validation and PDU specific processing. This is relevant, for example, when an OAM PDU and the VOE have independent MEL levels.

Forcing the VOE to process OAM PDUs as data frames is done by asserting INDEPENDENT_MEL when mapping the frame flow to the VOE:

- Analyzer: ANA_CL:IPT:OAM_MEP_CFG.INDEPENDENT_MEL_ENA
- Rewriter: ES0.ACTION.INDEPENDENT_MEL_ENA.

Asserting this option in the VOE lookup causes all frames in the corresponding frame flow to be processed as data frames, i.e. bypassing the frame validation, PDU specific updates, and statistics.

The following description of the Tx and Rx validation assumes INDEPENDENT_MEL = 0.

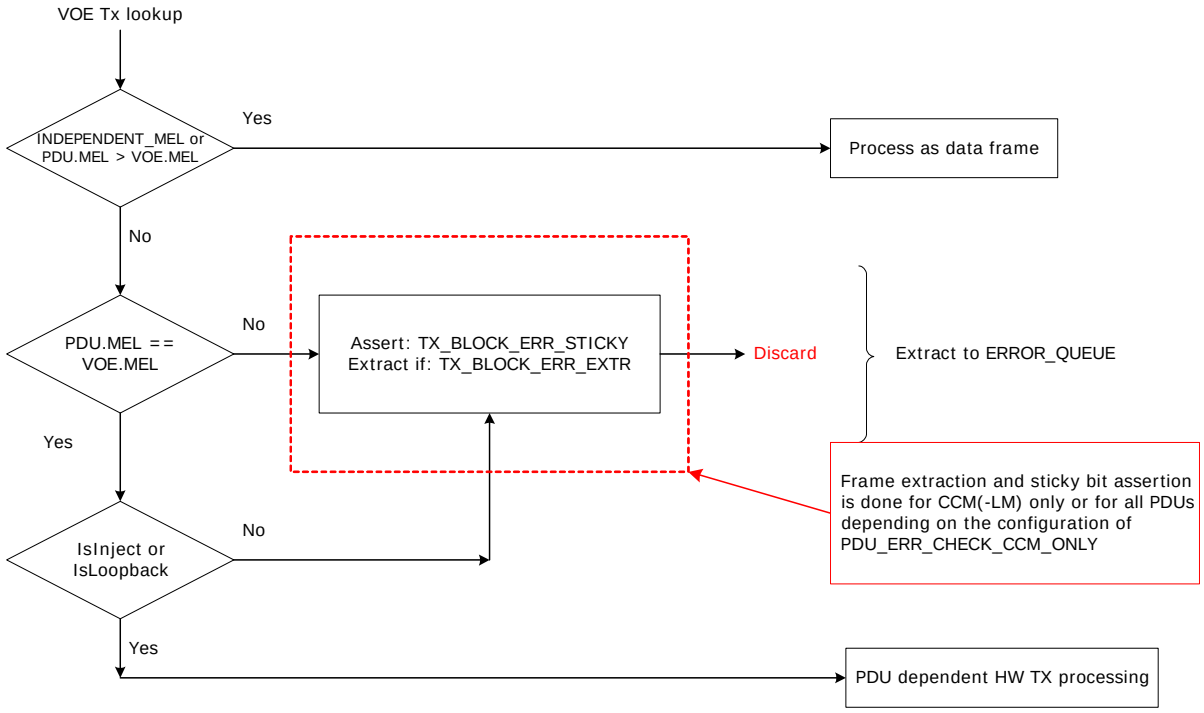
3.24.1.2 OAM PDU Validation: Tx Direction

When a frame is mapped to the VOE by a VOE Tx lookup, the VOE determines the frame type from IFH.DST.ENCAP.PDU_TYPE. The following frame types are supported by the Ethernet VOE:

- Y_1731 (Hardware-supported Y.1731 OAM PDUs)
- SAM_SEQ (Non-OAM sequence numbering. These frames are not valid Y.1731 PDUs.)

Only Y.1731 OAM PDUs are subject to the OAM PDU Tx validation shown in the following illustration. Other frame types are processed as data frames.

Figure 71 • **Ethernet VOE: Tx Validation**



OAM PDUs discarded by the Tx validation are counted in VOP:VOE_STAT:TX_OAM_DISCARD.TX_FRM_DISCARD_CNT. At the same time, the VOP:VOE_STAT:OAM_TX_STICKY.TX_BLOCK_ERR_STICKY bit is asserted, and the frames can optionally (VOP.VOE_SYAY:PDU_EXTRACT.TX_BLOCK_ERR_EXTR) be extracted to the CPU_ERR_QU.

It is configurable which OAM PDUs cause assertion of the VOP:VOE_STAT:OAM_TX_STICKY.TX_BLOCK_ERR_STICKY bit and extraction to the CPU. There are two possibilities:

- Only CCM(-LM) PDUs cause sticky bit assertion and extraction to the CPU.
- All PDUs cause sticky bit assertion and extraction to the CPU.

The PDUs that are extracted and assert the sticky bit are configured in VOP:VOE_CONF:OAM_CPU_COPY_CTRL.PDU_ERR_EXTRACT_CCM_ONLY. This configuration is common to the Tx and Rx direction.

If the frame passes the Tx validation, it is considered valid, and it is forwarded to PDU specific processing.

3.24.1.3 OAM PDU Validation: Rx Direction

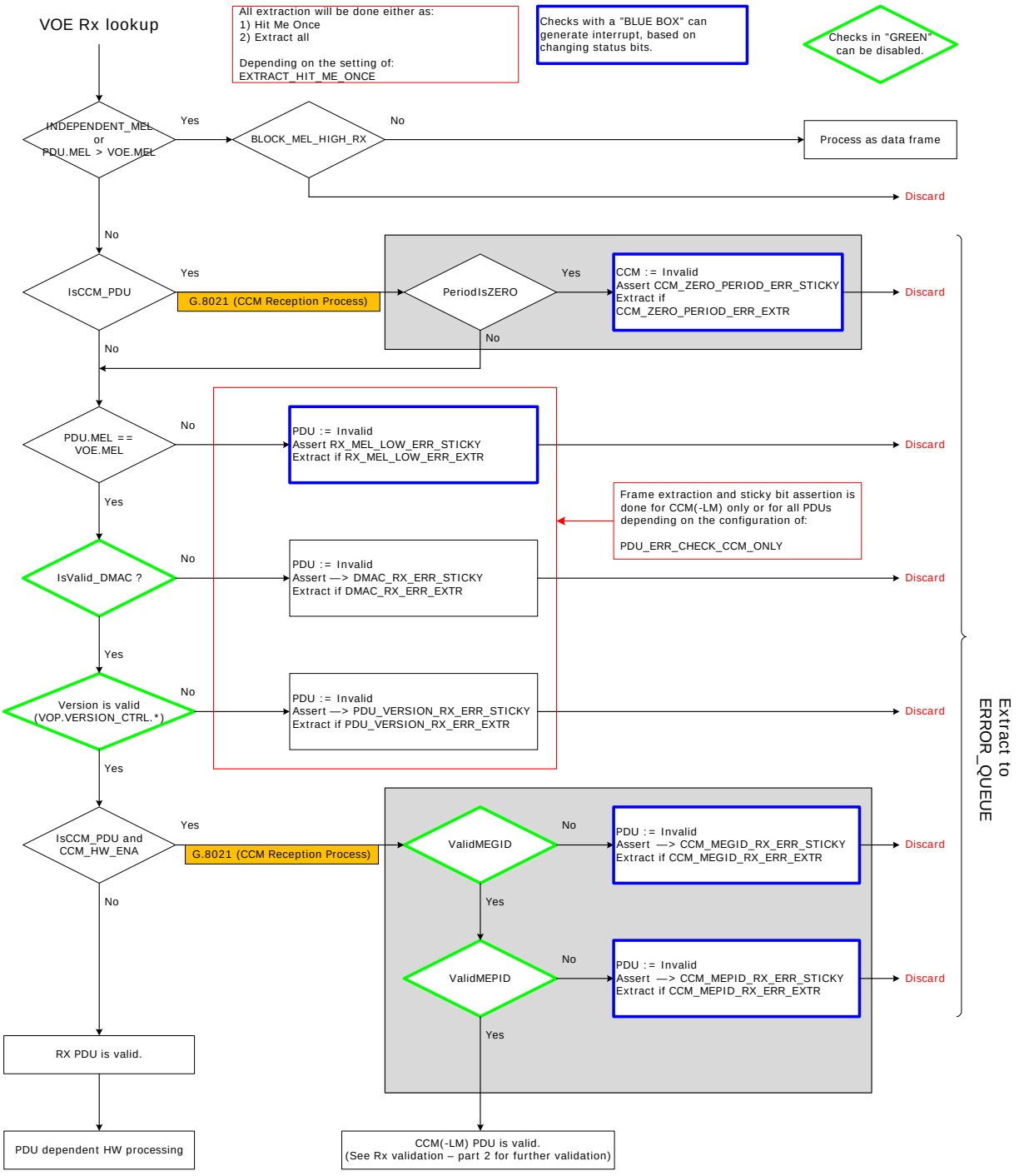
When a frame is mapped to the VOE by an VOE Rx lookup, the VOE determines whether this is a frame type (IFH.DST.ENCAP.PDU_TYPE) supported by the Ethernet VOE. Only Y.1731 OAM PDUs are subject to the OAM PDU Rx validation. Other frame types are processed as data frames.

The Rx validation is based on the following:

- MEL filtering (Y.1731)
- CCM(-LM) Rx (G.8021 section 8.1.7.3: CCM Reception Process)
- DMAC/Version validation

The following illustration shows the first part of Rx validation, which applies to all frames.

Figure 72 • **Rx Validation, Part 1**



The following describes the Rx validation of an OAM PDU.

The MEL validation is done in two parts. If the first check of OAM PDU.MEL > VOE.MEL fails, there is an extra check done only for CCM(-LM) frames. If BLOCK_MEL_HIGH_RX is 1, the frame is invalid and discarded. If BLOCK_MEL_HIGH_RX is 0, the OAM PDU is processed as a data frame.

If the OAM PDU.MEL is not higher than the VOE.MEL, a dedicated check is performed for CCM(-LM) PDUs prior to further MEL processing. If the CCM(-LM).Period is zero, it is an illegal frame and the CCM(-LM) is discarded. For every CCM(-LM) received, the VOE stores the result of the last zero period validation in VOP:VOE_STAT:CCM_RX_LAST.CCM_ZERO_PERIOD_ERR. If the state of the CCM(-LM)

zero period state changes a sticky bit is asserted (VOP:VOE_STAT:INTR_STICKY.CCM_ZERO_PERIOD_RX_ERR_STICKY). This can optionally generate an interrupt (VOP:VOE_STAT:INTR_ENA.CCM_ZERO_PERIOD_INTR_ENA). The illegal CCM(-LM) PDU can optionally be extracted to the CPU_ERR_QU (VOP:VOE_STAT:PDU_EXTRACT.CCM_ZERO_PERIOD_RX_ERR_EXTR).

If the OAM PDU.MEL == VOE.MEL, the OAM PDU is passed on for DMAC validation.

If the OAM PDU.MEL < VOE.MEL, the OAM PDU is marked as invalid. This implies that the OAM PDU is blocked, and the VOP:VOE_STAT:OAM_RX_STICKY.RX_MEL_LOW_BLOCK_STICKY bit is asserted.

For every CCM(-LM) received, the VOE stores the result of the last MEL check in VOP:VOE_STAT:CCM_RX_LAST.CCM_RX_MEL_LOW_ERR. If the value of this register changes, a sticky bit is asserted (VOP:VOE_STAT:INTR_STICKY.CCM_RX_MEL_LOW_ERR_STICKY). This can optionally generate an interrupt (VOP:VOE_STAT:INTR_ENA.CCM_RX_MEL_LOW_INTR_ENA). Any PDU failing the MEL check can optionally be extracted to the CPU_ERR_QU (VOP:VOE_STAT:PDU_EXTRACT.RX_MEL_LOW_ERR_EXTR). The VOE counts the number of CCM(-LM) discarded due to MEL filtering in VOP:VOE_STAT:CCM_ERR.CCM_RX_MEL_ERR_CNT.

The DMAC of the OAM PDU is checked according to VOP:VOE_CONF:VOE_CTRL.RX_DMAC_CHK_SEL. If the DMAC validation fails, the OAM PDU is marked as invalid. It is blocked and a sticky bit is asserted (VOP:VOE_STAT:OAM_RX_STICKY.DMAC_RX_ERR_STICKY). OAM PDUs failing the Rx DMAC validation can optionally be extracted to the CPU_ERR_QU (VOP:VOE_STAT:PDU_EXTRACT.DMAC_RX_ERR_EXTR).

Version validation is disabled for the following frames, because these frames are only processed by software.

- LTM/LTR
- Generic PDUs
- Unknown PDUs

Version validation of known OAM PDUs is enabled in VOP:VOE_CONF:VOE_CTRL.VERIFY_VERSION_ENA. If version validation is enabled, the VOE validates the version of known OAM PDUs against the versions configured for each PDU in the VOP::VERSION_CTRL.* and VOP::VERSION_CTRL_2.* registers.

The registers allow for individual configurations for each OAM PDU of which versions are accepted by the VOE. If the version of the OAM PDU is not a valid, the frame is blocked, and a sticky bit is asserted (VOP:VOE_STAT:OAM_RX_STICKY.PDU_VERSION_RX_ERR_STICKY). The OAM PDU can optionally be extracted to the CPU_ERR_QU (VOP:VOE_STAT:PDU_EXTRACT.PDU_VERSION_RX_ERR_EXTR).

If the version of the PDU is valid, further Rx validation depends on the OAM PDU type:

- CCM(-LM) PDUs are further validated
- SLM/SLR/1SL PDUs are passed on for SynLM validation. For more information, see SynLM PDU Rx Validation, page 257.
- Other PDUs are considered to be valid at this point and are forwarded for OAM PDU specific processing.

For CCM(-LM) PDUs, two PDU specific checks are done if CCM processing is enabled:

- Valid MEG-ID. The MEG-ID of incoming CCM(-LM) PDUs can optionally be validated (VOP:VOE_CONF:CCM_CFG.CCM_MEGID_CHK_ENA). The value of the incoming 48-byte CCM.MEG-ID is checked against VOP:VOE_CONF:CCM_MEGID_CFG.*.
  The result of the last MEG-ID validation is stored in VOP:VOE_STAT:CCM_RX_LAST.CCM_MEGID_ERR. If the value changes, the sticky bit VOP:VOE_STAT:INTR_STICKY.CCM_MEGID_RX_ERR_STICKY is asserted. This can optionally generate an interrupt (VOP:VOE_STAT:INTR_ENA.CCM_MEGID_INTR_ENA).
  If the MEG-ID validation fails, the CCM(-LM) PDU is invalid and it is discarded. It can optionally be extracted to the CPU error queue (VOP:VOE_STAT:PDU_EXTRACT.CCM_MEGID_RX_ERR_EXTR). The VOE counts the number of

Rx CCM(-LM) PDUs with MEGID error in VOP:VOE_STAT:CCM_RX_ERR_1.CCM_RX_MEGID_ERR_CNT.

- Valid MEP-ID. The MEP-ID of incoming CCM(-LM) PDUs can optionally be validated (VOP:VOE_CONF:CCM_CFG.CCM_MEPID_CHK_ENA). The value of the incoming CCM.MEP-ID is validated against the value in VOP:VOE_CONF:PEER_MEPID_CFG.PEER_MEPID.
  The result of the last MEP-ID validation is stored in VOP:VOE_STAT:CCM_RX_LAST.CCM_MEPID_ERR. If the value of this register changes, the sticky bit VOP:VOE_STAT:INTR_STICKY.CCM_MEPID_RX_ERR_STICKY is asserted. This can optionally generate an interrupt (VOP:VOE_STAT:INTR_ENA.CCM_MEPID_INTR_ENA).
  If the MEP-ID validation fails, the CCM(-LM) PDU is invalid and it is discarded. CCM(LM) PDUs with MEPID error are optionally extracted to the CPU error queue (VOP:VOE_STAT:PDU_EXTRACT.CCM_MEPID_RX_ERR_EXTR). The VOE counts the number of Rx CCM(-LM) PDUs with MEPID error in VOP:VOE_STAT:CCM_RX_ERR_1.CCM_RX_MEPID_ERR_CNT.

If the MEG-ID and MEP-ID are correctly validated, the CCM(-LM) PDU is considered a valid OAM PDU, and it is forwarded for hardware processing and clearing of the LOC counters.

It is configurable which PDUs failing the Rx validation checks (MEL, DMAC, version) causes the assertion of a sticky bit and can be extracted to the CPU:

- Only CCM(-LM) PDUs cause sticky bit assertion and extraction to the CPU.
- All PDUs cause sticky bit assertion and extraction to the CPU.

This is configured in VOP:VOE_CONF:OAM_CPU_COPY_CTRL.PDU_ERR_EXTRACT_CCM_ONLY. This configuration is common to the Tx and Rx direction.

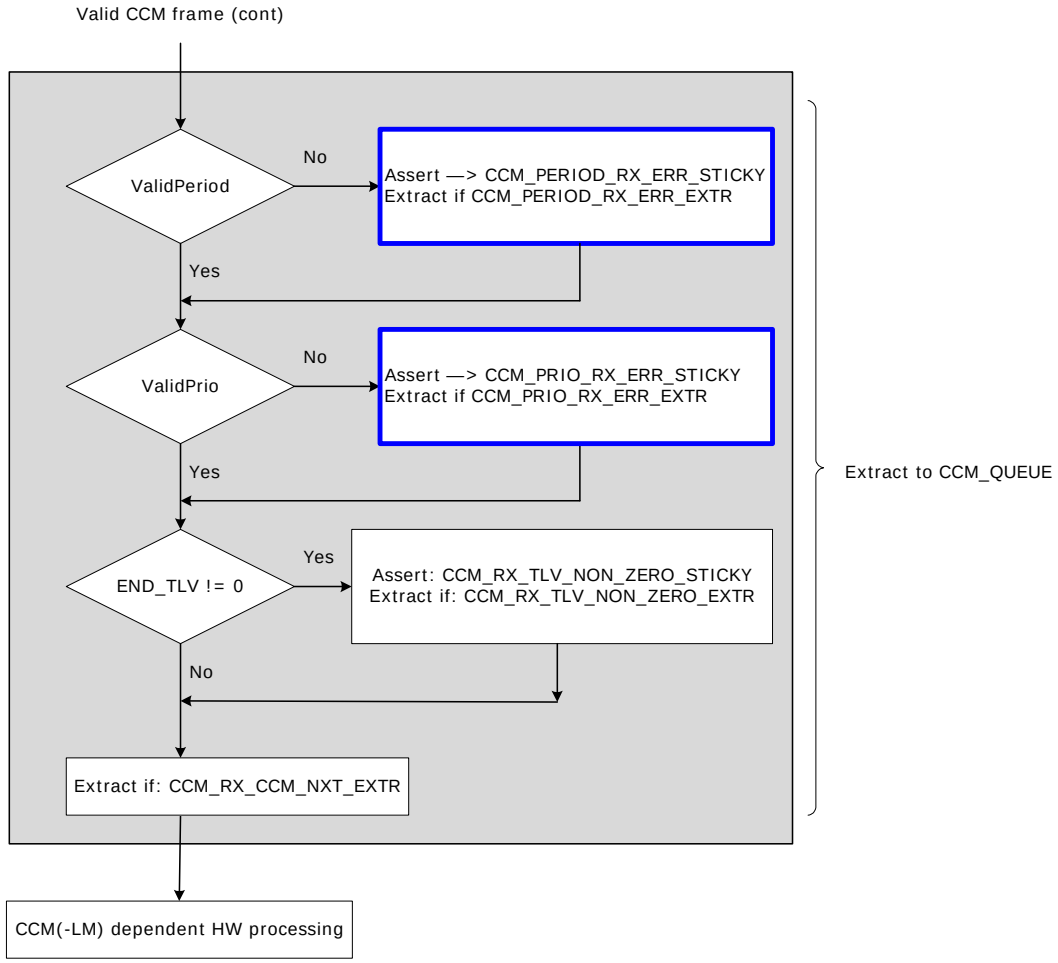
There are three additional checks done for valid CCM(-LM) PDUs. None of these tests invalidate the CCM(-LM) frame.

- CCM(-LM) period check
  The value of the CCM(-LM) period is validated against the value in VOP:VOE_CONF:CCM_CFG.CCM_PERIOD. The last result of this validation is stored in VOP:VOE_STAT:CCM_RX_LAST.CCM_PERIOD_ERR. If the value changes, the sticky bit VOP:VOE_STAT:INTR_STICKY.CCM_PERIOD_RX_ERR_STICKY is asserted. This can optionally generate an interrupt (VOP:VOE_STAT:INTR_ENA.CCM_PERIOD_INTR_ENA). The PDUs failing the period check can optionally be extracted to the CCM_CPU_QU queue (VOP:VOE_STAT:PDU_EXTRACT.CCM_PERIOD_RX_ERR_EXTR).

  The VOE counts the number of CCM(-LM) PDUs received with period error in VOP:VOE_STAT:CCM_RX_WARNING.CCM_RX_PERIOD_ERR_CNT.
- CCM(-LM) priority check
  The value of the CCM(-LM) priority is validated against the value in VOP:VOE_CONF:CCM_CFG.CCM_PRIO. The last result of this validation is stored in VOP:VOE_STAT:CCM_RX_LAST.CCM_PRIO_ERR. If the value changes, the sticky bit VOP:VOE_STAT:INTR_STICKY.CCM_PRIO_RX_ERR_STICKY is asserted. This can optionally generate an interrupt (VOP:VOE_STAT:INTR_ENA.CCM_PRIO_INTR_ENA). The PDUs failing the priority check can optionally be extracted to the CCM_CPU_QU queue (VOP:VOE_STAT:PDU_EXTRACT.CCM_PRIO_RX_ERR_EXTR).
- CCM(-LM) extra TLV check
  IEEE 802.1ag specifies that there can be TLVs appended to the CCM PDU in the CCM(-LM) frame. The VOE can process CCM(-LM) OAM PDUs with port status TLV or interface status TLV. For more information, see CCM TLV Processing, page 248. If a CCM(-LM) is received with any other kind of TLV, it is considered to have a non-zero END TLV. When the VOE receives a valid CCM(-LM) PDU with a non-zero END TLV field following the CCM(-LM) PDU and any of the TLVs described above, the VOP:VOE_STAT:OAM_RX_STICKY.CCM_RX_TLV_NON_ZERO_STICKY sticky bit is asserted. Valid CCM(-LM) PDUs with a non-zero END TLV after the CCM(-LM) PDU can optionally be extracted to the CCM_CPU_QU (VOP:VOE_STAT:PDU_EXTRACT.CCM_RX_TLV_NON_ZERO_EXTR). Note that this extraction is always hit-me-once.

The last part of the CCM(-LM) PDU specific Rx validation, which applies only to CCM and CCM-LM frames, is shown in the following illustration.

Figure 73 • **Rx Validation, Part 2**



PDUs that are discarded due to failing the Rx validation are counted in VOP:VOE_STAT:RX_OAM_DISCARD.RX_FRM_DISCARD_CNT.

3.24.1.4 Discarding Frames with High MEL

The VOE can be configured to terminate all Rx PDUs with MEL higher than the MEL configured for the VOE (VOP:VOE_CONF:VOE_CTRL.BLOCK_MEL_HIGH_RX).

If enabled, all Rx PDUs with MEL higher that the MEL configured for the VOE are discarded. Further the sticky bit VOP:VOE_STAT:OAM_RX_STICKY.RX_MEL_HIGH_BLOCK_STICKY is asserted. Frames discarded based on the above setting can be extracted to CPU queue VOP:VOE_STAT:PDU_EXTRACT.RX_MEL_HIGH_BLOCK_EXTR. PDUs discarded due this setting are counted in VOP:VOE_STAT:RX_OAM_DISCARD.RX_FRM_DISCARD_CNT.

3.24.1.5 Blocking Data Frames

The VOE can be configured to block all data frames in both Rx and Tx direction:

- Tx direction: VOP:VOE_CONF:VOE_CTRL.BLOCK_DATA_TX
- Rx direction: VOP:VOE_CONF:VOE_CTRL.BLOCK_DATA_RX.

When asserted, only valid OAM PDUs are processed by the VOE, all other traffic is blocked. Frames discarded due this setting in the Rx direction are counted in VOP:VOE_STAT:RX_OAM_DISCARD.RX_FRM_DISCARD_CNT while frames discarded due this setting in the TX direction are counted in VOP:VOE_STAT:TX_OAM_DISCARD.TX_FRM_DISCARD_CNT.

3.24.1.6 Looping OAM PDUs

The VOE supports looping the following PDUs:

- LBM to LBR
- LMM to LMR
- DMM to DMR
- SLM to SLR
- NON_OAM_MSG to NON_OAM_RSP

When looping an OAM PDU, the VOE performs the following actions in addition to OAM PDU specific updates:

- Inject the response
- Update IFH properties of the OAM response.
- Update the OpCode in the response from message to response.
- Swap the MAC addresses in the response.

If looping is enabled for one of the listed OpCodes, the VOE loops the incoming OAM Message to generate the corresponding OAM Response. This is done differently depending on whether the VOE is configured for Down- or Up-VOE:

- Looping in Down-VOE. The OAM Message is looped in the queuing system, which results in the OAM Response being injected into the egress path of the source port. This implies that when an OAM Message is looped by a Down-VOE, the corresponding OAM Response is transmitted unconditionally on the OAM Message Rx port.
- Looping in an Up-VOE. The OAM Message is looped in the LBK, which results in the OAM Response being injected into the ingress path of the VOE residence port. This implies that the OAM Response is forwarded based on IFH frame properties, based on classification and processing of the OAM Message at the ingress port with the exceptions described in the following.

When an OAM Message is looped, the OAM Response is injected with the same frame properties as the OAM Message with the following exceptions:

- VOE replaces the ISDX of the OAM response (VOP:VOE_CONF:LOOPBACK_CFG.LB_ISDX)
- VOE optionally clears the DP bit (VOP:VOE_CONF:LOOPBACK_CFG.CLEAR_DP_ON_LOOP)
- VOE optionally sets IFH.ES0_ISDX_ENA (VOP:VOE_CONF:LOOPBACK_CFG.LB_ES0_ISDX_ENA)

The VOE updates the OAM OpCode in the PDU:

- LBM (OpCode = 3) to LBR (OpCode = 2)
- LMM (OpCode = 43) to LMR (OpCode = 42)
- DMM (OpCode = 47) to DMR (OpCode = 46)
- SLM (OpCode = 55) to SLR (OpCode = 54)

The VOE modifies the frame MAC addresses as follows:

- OAM Response DMAC = OAM Message SMAC
- OAM Response SMAC = VOE Unicast Address

For more information about configuring VOE unicast address, see Basic Ethernet VOE Setup, page 245.

3.24.1.7 Basic Ethernet VOE Setup

The VOE is enabled in VOP:VOE_CONF:VOE_CTRL.VOE_ENA. If the VOE is not enabled, it can be configured, but no frame validation and no frame processing is done.

The VOE MEG level (MEL) is configured in VOP:VOE_CONF:VOE_CTRL.MEL_VAL. This value is used for MEL filtering of OAM PDUs in both the Rx and Tx direction.

The VOE is either configured as an Up-VOE or a Down-VOE. This is configured in VOP:VOE_CONF:VOE_CTRL.UPMEP_ENA.

As part of the Rx OAM PDU validation, the VOE validates the DMAC of the received OAM PDU. The DMAC Rx check is configured in VOP:VOE_CONF:VOE_CTRL.RX_DMAC_CHK_SEL. The DMACs used for validation are either a common multicast address or a VOE unicast address. The MAC addresses are configured in:

- Multicast:
  VOP::COMMON_MEP_MC_MAC_MSB.MEP_MC_MAC_MSB
  VOP::COMMON_MEP_MC_MAC_LSB.MEP_MC_MAC_LSB.
- Unicast:
  VOP:VOE_CONF:MEP_UC_MAC_MSB.MEP_UC_MAC_MSB
  VOP:VOE_CONF:MEP_UC_MAC_LSB.MEP_UC_MAC_LSB.

3.24.1.8 OAM PDU Counters

Each VOE includes a set of OAM counters to count the number of OAM PDUs processed. Each PDU type is configured in VOP:VOE_CONF.OAM_CNT_OAM_CTRL as either selected PDU type or non-selected PDU type.

- Selected Tx PDUs are counted in VOP:VOE_STAT:TX_SEL_OAM_CNT.TX_SEL_OAM_FRM_CNT.
- Non-selected Tx PDUs are counted in VOP:VOE_STAT:TX_OAM_FRM_CNT.TX_OAM_FRM_CNT.
- Selected Rx PDUs are counted in VOP:VOE_STAT:RX_SEL_OAM_CNT.RX_SEL_OAM_FRM_CNT.
- Non-selected Rx PDUs are counted in VOP:VOE_STAT:RX_OAM_FRM_CNT.RX_OAM_FRM_CNT.

3.24.1.9 SAM per COSID Sequence Numbering

The default VOE configuration supports common sequence numbering across all COSIDs for the following PDUs:

- TST
- LBM/LBR
- CCM(-LM)
- SAM_SEQ

The VOP allows 32 VOEs to be configured for per COSID sequence numbering. This is done by implementing sequence numbering for COSID = 7 in the existing VOE_STAT register, while implementing support for COSID = 0 through 6 in VOP:SAM_COSID_SEQ_CNT.

To enable per COSID sequence numbering for TST, LBM/LBR or SAM_SEQ, set VOP:VOE_CONF:SAM_COSID_SEQ_CFG.PER_COSID_LBM = 1. To enable per COSID sequence numbering for CCM-LM, set VOP:VOE_CONF:SAM_COSID_SEQ_CFG.PER_COSID_CCM = 1. Only one of the above can be enabled for a VOE.

When per COSID sequence numbering is enabled, the VOE must be assigned a per COSID counter set in VOP:VOE_CONF:SAM_COSID_SEQ_CFG.PER_COSID_CNT_SET. This configuration causes the counters in VOP:SAM_COSID_SEQ_CNT to be updated for COSID 0 through 6.

Configuring a VOE for per COSID sequence numbering only affects the sequence numbering, all other VOE functionality is unaffected.

3.24.2 Continuity Check Messages (CCM)

This section describes processing of valid CCM(-LM).

3.24.2.1 CCM and CCM-LM Frames

Within the scope of the VOE, a CCM-LM frame is defined as a CCM PDU with one or more non-zero LM counters in the payload. The VOE processes CCM-LM PDUs in the same way it processes CCM PDUs. The functionality for CCM PDUs also holds true for CCM-LM PDUs. For more information about how the VOE processes the F-LM contents of CCM-LM PDUs, see Frame Loss Measurement, Dual-Ended, page 253.

The VOE classifies and processes a PDU as a CCM-LM frame if the following is true:

- CCM processing is enabled (VOP:VOE_CONF:OAM_HW_CTRL.CCM_ENA).
- CCM PDU contains one or more non-zero LM counter in payload.
- CCM-LM processing is enabled (VOP:VOE_CONF:OAM_HW_CTRL.CCM_LM_ENA).

If CCM-LM is not enabled, any CCM-LM PDUs received by the VOE is processed as CCM PDUs. Classification of CCM-LM PDUs is the same in the Tx and Rx direction.

3.24.2.2 Enabling CCM Hardware Processing

The processing of CCM PDUs is enabled in VOP:VOE_CONF:OAM_HW_CTRL.CCM_ENA. If CCM hardware processing is not enabled, a CCM PDU is not updated and parts of the CCM(-LM) Rx validation are bypassed. For more information, see OAM PDU Validation: Rx Direction, page 240.

3.24.2.3 CCM(-LM) Counter Configuration

The OAM PDU Tx/Rx counters are updated according to the counter configuration in CCM: VOP:VOE_CONF:OAM_CNT_OAM_CTRL.CCM_OAM_CNT_ENA and CCM-LM: VOP:VOE_CONF:OAM_CNT_OAM_CTRL.CCM_LM_OAM_CNT_ENA.

CCM(-LM) PDUs are optionally included in the F-LM counters. For more information, see Frame Loss Measurement (F-LM), page 231.

- CCM(-LM): VOP:VOE_CONF:OAM_CNT_DATA_CTRL.CCM_DATA_CNT_ENA

3.24.2.4 CCM(-LM) Frame Tx

The VOE Tx lookup for a valid CCM(-LM) PDU includes the following actions:

- Optionally update CCM PDU sequence number in VOP:VOE_CONF:CCM_CFG.CCM_SEQ_UPD_ENA. The VOE updates the CCM PDU sequence number with the configured value (VOP:VOE_STAT:CCM_TX_SEQ_CFG.CCM_TX_SEQ).
- Increment the value of VOP:VOE_STAT:CCM_TX_SEQ_CFG.CCM_TX_SEQ by 1. This is also done when CCM_SEQ_UPD_ENA = 0 to allow counting CCM frames.
- The value of the CCM(-LM).RDI field is overwritten with the configured value in VOP:VOE_STAT:CCM_STAT.CCM_TX_RDI.

3.24.2.5 CCM(-LM) Frame Rx

The VOE Rx lookup for a valid CCM(-LM) PDU includes the following actions:

- Updating CCM Rx sticky bit (VOP:VOE_STAT:OAM_RX_STICKY.CCM_RX_STICKY).
- Updating CCM-LM Rx sticky bit (VOP:VOE_STAT:OAM_RX_STICKY.CCM_LM_RX_STICKY).
- The VOE optionally validates that the CCM(-LM) PDU contains the expected sequence number (VOP:VOE_CONF:CCM_CFG.CCM_RX_SEQ_CHK_ENA). The VOE validates the CCM PDU sequence number against the value VOP:VOE_STAT:CCM_RX_SEQ_CFG.CCM_RX_SEQ + 1. If the sequence check fails, the VOP:VOE_STAT:OAM_RX_STICKY.CCM_RX_SEQ_ERR_STICKY bit is asserted. The number of CCM(-LM) with out-of-sequence errors are counted in VOP:VOE_STAT:CCM_RX_WARNING.CCM_RX_SEQNO_ERR_CNT.
- Stores the received CCM(-LM) PDU sequence in VOP:VOE_STAT:CCM_RX_SEQ_CFG.CCM_RX_SEQ.
- The value of the PDU.RDI value is compared to the stored value in VOP:VOE_STAT:CCM_RX_LAST.CCM_RX_RDI. If the value differs, the VOP:VOE_STAT:INTR_STICKY.CCM_RX_RDI_STICKY bit is asserted. When the sticky bit is asserted, an interrupt is optionally generated in VOP:VOE_STAT:INTR_ENA.CCM_RX_RDI_INTR_ENA.The value of the CCM PDU.RDI bit is stored in VOP:VOE_STAT:CCM_RX_LAST.CCM_RX_RDI.
- The VOE CCM miss counter is cleared (VOP:VOE_STAT:CCM_STAT.CCM_MISS_CNT := 0).
- If the VOE was in LOC state indicated by VOP:VOE_STAT:CCM_RX_LAST.CCM_LOC_DEFECT = 1, the VOP:VOE_STAT:INTR_STICKY.CCM_LOC_STICKY bit is asserted. This can optionally generate an interrupt (VOP:VOE_STAT:INTR_ENA.CCM_LOC_INTR_ENA. The VOE clears the LOC state (asserted by the LOC scan) in VOP:VOE_STAT:CCM_RX_LAST.CCM_LOC_DEFECT := 0.

The VOE contains the CCM(-LM) specific Rx counters:

- Valid CCM(-LM) PDUs: VOP:VOE_STAT:CCM_RX_FRM_CNT.CCM_RX_VLD_FC_CNT
- In-valid CCM(-LM) PDUs: VOP:VOE_STAT:CCM_RX_FRM_CNT.CCM_RX_INVLD_FC_CNT

The VOE optionally extracts the next valid CCM(-LM) PDUs through VOP:VOE_STAT:PDU_EXTRACT.CCM_RX_CCM_NXT_EXTR.

The VOE optionally extracts all valid Rx CCM(-LM) PDUs:

- CCM: VOP:VOE_CONF:OAM_CPU_COPY_CTRL.CCM_CPU_COPY_ENA
- CCM-LM: VOP:VOE_CONF:OAM_CPU_COPY_CTRL.CCM_LM_CPU_COPY_ENA

3.24.2.6 CCM Source Move Detect

The VOE supports CCM source port detection to determine when CCM frames arrive from a new source port, for instance after a network failover in a protected ring. In case the Rx port is a LAG, the physical port number of the LAG is used for the detection.

The source port of the previous incoming CCM PDU is stored in VOP:VOE_STAT:CCM_RX_LAST.CCM_RX_SRC_PORT.

The number of consecutive CCM PDUs received on this port are counted in VOP:VOE_STAT:CCM_STAT.CCM_RX_SRC_PORT_CNT.

The source port is considered to be stable, when the number of consecutive CCM PDUs received on this port reaches a configurable level in VOP::VOP_CTRL.CCM_RX_SRC_PORT_DETECT_CNT. When the source port is stable, the VOP:VOE_STAT:INTR_STICKY.CCM_RX_SRC_PORT_DETECT_STICKYbit is asserted. The sticky bit can optionally generate an interrupt through VOP:VOE_STAT:INTR_ENA.CCM_RX_SRC_PORT_DETECT_INTR_ENA.

If the source port changes, the CCM source port counter is reset to zero.

3.24.3 VOE LOCC Configuration

The CCM(-LM) functionality in the VOE interacts with the LOCC in two ways: CCM miss counter and CCM-LM insertion timer.

3.24.3.1 CCM TLV Processing

The VOE can process two types of CCM(-LM) TLVs: port status and interface status.

The VOE can process one or both of the TLVs in any order, within the same CCM(-LM) frame, provided that the TLVs are located as the first TLV after the CCM(-LM) PDU.

If another TLV is found either before the port status TLV or the interface status TLV, the VOE will stop processing the TLVs and consider this a CCM(-LM) PDU with a non-zero END TLV. Such frames will assert a sticky bit and can optionally be extracted to the CPU. For more information, see OAM PDU Validation: Rx Direction, page 240.

When a valid CCM(-LM) is received by the VOE, which contains a port status TLV or an interface status TLV, or both, the VOE will sample the value of the port status or interface status (or both) and store the value. If a value changes, the VOE can optionally assert an interrupt.

The following registers control port status TLV processing.

- VOP:VOE_STAT:CCM_RX_LAST.TLV_PORT_STATUS
- VOP:VOE_STAT:INTR_STICKY.TLV_PORT_STATUS_STICKY
- VOP:VOE_STAT:INTR_ENA.TLV_PORT_STATUS_INTR_ENA

The following registers control the interface status TLV processing.

- VOP:VOE_STAT:CCM_RX_LAST.TLV_INTERFACE_STATUS
- VOP:VOE_STAT:INTR_STICKY.TLV_INTERFACE_STATUS_STICKY
- VOP:VOE_STAT:INTR_ENA.TLV_INTERFACE_STATUS_INTR_ENA

3.24.3.2 CCM Miss Counter

To detect an LOC condition, the VOE implements a 3-bit CCM miss counter in VOP:VOE_STAT:CCM_STAT.CCM_MISS_CNT that is incremented by the LOCC and cleared every time a valid CCM(-LM) is received.

The VOE is assigned an LOC timer in VOP:VOE_CONF:CCM_CFG.CCM_PERIOD. The CCM miss counter is incremented every time the assigned LOC timer expires. This is the same register that is used to validate the period of valid Rx frames. For more information about CCM validation, see OAM PDU Validation: Rx Direction, page 240.

When the CCM miss counter equals 7, a loss of continuity (LOC) defect event is signaled by the VOE. The current LOC defect state can be read in the VOP:VOE_STAT:CCM_RX_LAST.CCM_LOC_DEFECT status bit. When the state of the status bit changes, the VOP:VOE_STAT:INTR_STICKY.CCM_LOC_STICKY bit is asserted. When this sticky bit is asserted, an interrupt is optionally asserted (VOP:VOE_STAT:INTR_ENA.CCM_LOC_INTR_ENA). The VOE generates an interrupt if the assigned LOC timer expires seven times without reception of a valid CCM(-LM) frame. This is used to generate an interrupt after 3.5 of the configured CCM periods, as specified by Y.1731. The status of the VOE interrupt can be read in VOP::INTR.

3.24.3.3 CCM-LM Insertion Timer

The VOE converts the next valid Tx CCM PDU into a CCM-LM by inserting LM information into the payload when VOP:VOE_STAT:CCM_STAT.CCM_LM_INSERT_NXT is set. When inserting the LM information into a CCM frame (CCM-LM insertion), the bit is automatically cleared by hardware. This register can be set by the CPU or automatically by the LOCC by assigning an LOC timer, which asserts the signal every time it expires (VOP:VOE_CONF:CCM_CFG.CCM_LM_PERIOD).

If the AFI is configured to generate CCM PDUs at a fast rate, 3.3 ms for example, this mechanism can be used to insert LM contents into one of these CCM PDUs every 100 ms.

3.24.4 Test Frames (TST)

Test frames share VOE resources with the LBM/LBR and SAM_SEQ OAM PDUs, so only one of these PDU types can be active in the VOE at any given time. TST PDUs can be enabled for per COSID sequence numbering. For more information, see SAM per COSID Sequence Numbering, page 246.

3.24.4.1 Enabling TST Hardware Processing

The processing of TST PDUs is enabled in VOP:VOE_CONF:OAM_HW_CTRL.TST_ENA. If TST hardware processing is not enabled, only statistics are updated, no modifications are done to the TST PDU.

3.24.4.2 TST Counter Configuration

The OAM PDU Tx/Rx counters are updated according to the counter configuration in VOP:VOE_CONF:OAM_CNT_OAM_CTRL.TST_OAM_CNT_ENA.

It is optional by configuration in VOP:VOE_CONF:OAM_CNT_DATA_CTRL.TST_DATA_CNT_ENA whether valid TST PDUs are included in the F-LM counters described in section Frame Loss Measurement (F-LM), page 231.

3.24.4.3 TST Frame Tx

The VOE Tx lookup for a valid TST PDU includes the following actions:

- The VOE optionally updates the TST PDU sequence number (VOP:VOE_CONF:TX_TRANSID_UPDATE.TST_UPDATE_ENA).
- TST PDU sequence number is updated with the stored value (VOP:VOE_STAT:LBM_TX_TRANSID_CFG.LBM_TX_TRANSID).
- Update value of the above listed register by +1.

3.24.4.4 TST Frame Rx

The VOE Rx lookup for a valid TST PDU includes the following actions:

- Updating TST Rx sticky bit (VOP:VOE_STAT:OAM_RX_STICKY.TST_RX_STICKY).
- Validating the TST PDU transaction ID number against the expected value (VOP:VOE_STAT:LBR_RX_TRANSID_CFG.LBR_RX_TRANSID + 1).
- If the check fails, the VOP:VOE_STAT:OAM_RX_STICKY.LBR_TRANSID_ERR_STICKY bit is asserted.
- Saving the received TST PDU sequence number in the above listed register.
- The VOE counts the number of valid Rx TST PDUs (VOP:VOE_STAT:LBR_RX_FRM_CNT.LBR_RX_FRM_CNT).
- The VOE counts the number of TST PDUs received with sequence errors (VOP:VOE_STAT:LBR_RX_TRANSID_ERR_CNT.LBR_RX_TRANSID_ERR_CNT).

- The VOE optionally extracts the next valid TST PDU (VOP:VOE_STAT:PDU_EXTRACT.RX_TEST_FRM_NXT_EXTR).
- The VOE optionally extracts all valid Rx TST PDUs (VOP:VOE_CONF:OAM_CPU_COPY_CTRL.TST_CPU_COPY_ENA).

3.24.4.5 TST TLV CRC Check

Y.1731 specifies that the TST PDU may include a TEST TLV. The TEST TLV includes a data field and an optional CRC field, which covers the data. The VOE optionally validates this CRC field in all valid TST PDUs carrying a TLV and counts the number of Rx TST PDUs that fail the CRC check. CRC checking requires the TEST TLV to be the first TLV following the TST PDU, and only a single TEST TLV can be verified. There is no support for inserting the TEST TLV and generating the CRC value. This must be done outside the VOE.

The validation of the CRC value in the TEST TLV is enabled in VOP:VOE_CONF:OAM_HW_CTRL.TST_TLV_CRC_VERIFY_ENA.

The number of TST PDUs received with TLV CRC errors is counted in VOP:VOE_CRC_ERR:LBR_CRC_ERR_CNT.LBR_CRC_ERR_CNT.

Y.1731 specifies that the CRC field in the TEST TLV is valid for the pattern type 1 and type 3. Hence the VOE only validates the TLV CRC field for these test pattern types.

3.24.5 Loopback Frames (LBM/LBR)

LBM/LBR share VOE resources with the OAM PDUs TST and SAM_SEQ, so only one of these PDU types can be active in the VOE at any given time. LBM/LBR PDUs can be enabled for per COSID sequence numbering. For more information, see SAM per COSID Sequence Numbering, page 246.

For information about special functions for LBM/LBR processing when the VOE is enabled for G.8113.1 OAM processing, see G.8113.1 Specific Functions, page 272.

3.24.5.1 Enabling LBM/LBR Hardware Processing

The processing of LBM PDUs is enabled in VOP:VOE_CONF:OAM_HW_CTRL.LBM_ENA and LBR PDUs in VOP:VOE_CONF:OAM_HW_CTRL.LBR_ENA. If LBM/LBR hardware processing is not enabled, only statistics are updated; no modifications are made to the LBM/LBR PDU.

3.24.5.2 LBM/LBR Counter Configuration

The OAM PDU Tx/Rx counters are updated according to the counter configuration:

- LBM: VOP:VOE_CONF:OAM_CNT_OAM_CTRL.LBM_OAM_CNT_ENA
- LBR: VOP:VOE_CONF:OAM_CNT_OAM_CTRL.LBR_OAM_CNT_ENA

Valid LBM/LBR PDUs are optionally included in the F-LM counters:

- LBM: VOP:VOE_CONF:OAM_CNT_DATA_CTRL.LBM_DATA_CNT_ENA
- LBR: VOP:VOE_CONF:OAM_CNT_DATA_CTRL.LBR_DATA_CNT_ENA

3.24.5.3 LBM Frame Tx

The VOE Tx lookup for a valid LBM PDU includes the following actions:

- Optionally updating the LBM sequence number (VOP:VOE_CONF:TX_TRANSID_UPDATE.LBM_UPDATE_ENA).
- Updating the LBM sequence number with the stored value (VOP:VOE_STAT:LBM_TX_TRANSID_CFG.LBM_TX_TRANSID).
- Incrementing the value of the above listed register by +1.

3.24.5.4 LBM Frame Rx

The VOE Rx lookup for a valid LBM PDU includes the following actions:

- Updating the LBM Rx sticky bit (VOP:VOE_STAT:OAM_RX_STICKY.LBM_RX_STICKY).
- The VOE optionally loops valid LBM PDUs (VOP:VOE_CONF:LOOPBACK_ENA.LB_LBM_ENA).If loopback is enabled, an LBR PDU is injected by the VOE in the Tx direction towards the peer MEP. The injected LBR is processed by VOE Tx lookup.

- The VOE optionally extracts all valid Rx LBM PDUs (VOP:VOE_CONF:OAM_CPU_COPY_CTRL.LBM_CPU_COPY_ENA).

3.24.5.5 LBR Frame Tx

The VOE Tx lookup for a valid LBR PDU counts the number of valid Tx LBR PDUs (VOP:VOE_STAT:LBR_TX_FRM_CNT.LBR_TX_FRM_CNT).

3.24.5.6 LBR Frame Rx

The VOE Rx lookup for a valid LBM PDU includes the following actions:

- Asserting the LBR Rx sticky bit (VOP:VOE_STAT:OAM_RX_STICKY.LBR_RX_STICKY).
- Validating the LBR PDU transaction ID number against the expected value (VOP:VOE_STAT:LBR_RX_TRANSID_CFG.LBR_RX_TRANSID + 1).
- If the check fails, the VOP:VOE_STAT:OAM_RX_STICKY.LBR_TRANSID_ERR_STICKY bit is asserted.
- Stores the received LBR PDU transaction ID in the above listed register.
- The VOE counts the number of valid Rx LBR PDUs (VOP:VOE_STAT:LBR_RX_FRM_CNT.LBR_RX_FRM_CNT). An LBR PDU is valid even when the transaction ID number differs from the expected.
- The VOE counts the number of LBR PDUs received with sequence errors (VOP:VOE_STAT:LBR_RX_TRANSID_ERR_CNT.LBR_RX_TRANSID_ERR_CNT).
- The VOE optionally extracts the next valid LBR PDU (VOP:VOE_STAT:PDU_EXTRACT.RX_TEST_FRM_NXT_EXTR).
- The VOE optionally extracts all valid Rx LBR PDUs (VOP:VOE_CONF:OAM_CPU_COPY_CTRL.LBR_CPU_COPY_ENA).

3.24.5.7 LBR TLV CRC Check

Y.1731 specifies that the LBR PDU may include a TEST TLV. The TEST TLV includes a data field and an optional CRC field that covers the data. This is exactly the same processing as for TST PDUs. For more information, see TST TLV CRC Check, page 250.

Verification of the Test TLV CRC for Rx LBR PDUs is optionally enabled in VOP:VOE_CONF:OAM_HW_CTRL.LBR_TLV_CRC_VERIFY_ENA. This verification is not supported if the VOE is configured for G.8113.1, because G.8113.1 specifies that the LBM/LBR PDUs must be immediately followed by a “target MEP/MIP ID” TLV (LBM) or a “replying MEP/MIP ID” TLV (LBR). This requirement prevents the TEST TLV from being the first TLV following the LBM/LBR PDU, which is required if the VOE is to verify the CRC of the TEST TLV.

3.24.6 Frame Loss Measurement, Single-Ended

The VOE supports two types of frame loss measurement specified by Y.1731:

- Single-ended loss measurement (SE-LM). SE-LM use LMM/LMR PDUs
- Dual-ended measurement (DE-LM). DE-LM use CCM-LM PDUs

The VOE counters are shared between SE-LM and DE-LM, so a VOE only supports one type of F-LM at any given time. Note that the type of F-LM running on a given connection depends only on the PDUs transmitted as part of the F-LM session. There is no specific VOE settings to configure single-ended versus dual-ended F-LM.

The F-LM PDUs for both single-ended and dual-ended LM unconditionally sample the LM counters indicated by IFH.COSID. In the following terms are used:

- Tx LM Counter: The VOE Tx F-LM counter indicated by the LMM/LMR IFH.COSID.
- Rx LM Counter: The VOE Rx F-LM counter indicated by the LMM/LMR IFH.COSID.

This section describes single-ended loss measurement. For information about dual-ended loss measurement, see Frame Loss Measurement, Dual-Ended, page 253.

Single-ended loss measurement uses LMM/LMR PDUs to sample the F-LM counters. For more information, see Frame Loss Measurement (F-LM), page 231.

The implementation of single ended frame loss measurement reuses VOE configuration and statistics resources used for Synthetic LM (SLM/SLR). This implies that a single VOE does not support F-LM

simultaneously with the processing of SynLM. If SLM/SLR PDUs are received on a VOE configured for F-LM, they assert the correct Rx sticky bit but the remaining processing is not guaranteed. The same applies if LMM/LMR PDUs are received on a VOE enabled for SynLM.

3.24.6.1 Enabling LMM/LMR Hardware Processing

The processing of LMM PDUs is enabled in VOP:VOE_CONF:OAM_HW_CTRL.LMM_ENA and LMR PDUs in VOP:VOE_CONF:OAM_HW_CTRL.LMR_ENA. If LMM/LMR hardware processing is not enabled, only statistics are updated, no modifications are done to the LMM/LMR PDU.

3.24.6.2 LMM/LMR Counter Configuration

The OAM PDU Tx/Rx counters are updated according to the counter configuration:

- LMM: VOP:VOE_CONF:OAM_CNT_OAM_CTRL.LMM_OAM_CNT_ENA
- LMR: VOP:VOE_CONF:OAM_CNT_OAM_CTRL.LMR_OAM_CNT_ENA

Valid LMM/LMR PDUs are optionally included in the F-LM counters:

- LMM: VOP:VOE_CONF:OAM_CNT_DATA_CTRL.LMM_DATA_CNT_ENA
- LMR: VOP:VOE_CONF:OAM_CNT_DATA_CTRL.LMR_DATA_CNT_ENA

3.24.6.3 LMM Frame Tx

The VOE Tx lookup for a valid LMM PDU includes the following actions:

- Updating LMM.TxFCf field with Tx LM counter.
- Counting the number of Tx LMM frames (VOP:VOE_STAT:LM_PDU_CNT.LMM_TX_PDU_CNT).

3.24.6.4 LMM Frame Rx

The VOE Rx lookup for a valid LMM PDU includes the following actions:

- Updating LMM Rx sticky bit (VOP:VOE_STAT:OAM_RX_STICKY.LMM_RX_STICKY).
- Updating LMM.RxFCf with Rx LM counter.
- The VOE optionally loops valid LMM PDUs (VOP:VOE_CONF:LOOPBACK_ENA.LB_LMM_ENA). If loopback is enabled, an LMR PDU is injected by the VOE in the Tx direction towards the peer MEP. The injected LMR is processed by VOE Tx lookup.
- Counting the number of Rx LMM frames (VOP:VOE_STAT:LM_PDU_CNT.LMM_RX_PDU_CNT).
- The VOE optionally extracts all valid Rx LMM PDUs (VOP:VOE_CONF:OAM_CPU_COPY_CTRL.LMM_CPU_COPY_ENA).

3.24.6.5 LMR Frame Tx

The VOE Tx lookup for a valid LMR PDU includes the following actions:

- Updating LMR.TxFCb with the Tx LM counter.
- Counting the number of Tx LMR frames (VOP:VOE_STAT:LM_PDU_CNT.LMR_TX_PDU_CNT).

3.24.6.6 LMR Frame Rx

The VOE Rx lookup for a valid LMR PDU updates LMR Rx sticky bit (VOP:VOE_STAT:OAM_RX_STICKY.LMR_RX_STICKY).

When a valid LMR PDU is received by the VOE, it samples the Rx LM counter denoted RxFCb. However, the LMR PDU format does not reserve any bits to hold this value. The VOE can be programmed to insert the RxFCb value in the 32 bits following the LMR.TxFCb through VOP::VOP_CTRL.LMR_UPD_RXFCL_ENA. When this option is enabled the following happens:

- LMR.TLV is altered from 12 to 16.
- The RxFCb value is written to the 32 bits following the LMR.TxFCb in the LMR PDU
- The byte following the newly written LMR.RxFCb constitutes the PDU end TLV. This byte is set to zero. Note that the these modifications are not in line with the Y.1731 standard, hence the resulting frame is not a standards compliant PDU. However, it is assumed that the frame is subsequently forwarded to an internal/external CPU. If inserting the RxFCb into the received LMR PDU is not enabled, it is not possible to extract the RxFCb value.
- Counting the number of Rx LMR frames (VOP:VOE_STAT:LM_PDU_CNT.LMR_RX_PDU_CNT).

- The VOE optionally extracts all valid Rx LMR PDUs (VOP:VOE_CONF:OAM_CPU_COPY_CTRL.LMR_CPU_COPY_ENA).

3.24.7 Frame Loss Measurement, Dual-Ended

Dual-ended LM uses CCM-LM PDUs to sample the F-LM counters. The CCM-LM PDUs unconditionally sample the counters indicated by the IFH.COSID. As a result, when running dual-ended LM, CCM frames must always be enabled, and all CCM-LM frames are processed as CCM PDUs in addition to the dual-ended LM functions described in the following.

For more information about F-LM counters, see Frame Loss Measurement (F-LM), page 231

3.24.7.1 Enabling CCM-LM Hardware Processing

The processing of CCM-LM PDUs is enabled in VOP:VOE_CONF:OAM_HW_CTRL.CCM_LM_ENA. If CCM-LM hardware processing is not enabled CCM-LM PDUs are processed as CCM PDUs. Only exception is that the Rx sticky bit is asserted, to allow for detection of CCM PDUs with unexpected LM information.

3.24.7.2 CCM-LM Counter Configuration

The OAM PDU Tx/Rx counters are updated according to the counter configuration in VOP:VOE_CONF:OAM_CNT_OAM_CTRL.CCM_LM_OAM_CNT_ENA.

Optionally, valid CCM-LM PDUS can be included in the F-LM counters by configuring VOP:VOE_CONF:OAM_CNT_DATA_CTRL.CCM_DATA_CNT_ENA.

3.24.7.3 CCM-LM Frame Injection

Because there is no separate Y.1731 OpCode for CCM-LM PDUs, another mechanism is required to inform the VOE to insert LM information into selected CCM PDUs. CCM-LM PDUs can be signaled to the VOE in two ways:

- Injecting a CCM PDU with non-zero LM counter fields. When the VOE detects a valid CCM PDU with one or more non-zero LM counter fields the PDU is classified as a CCM-LM PDU.
- Use VOE CCM-LM insertion. Asserting the following field causes the VOE to classify the next valid CCM PDU as a CCM-LM PDU (VOP:VOE_STAT:CCM_STAT.CCM_LM_INSERT_NXT). This field can be auto-asserted by the LOCC. For more information, see CCM-LM Insertion Timer, page 249.

3.24.7.4 CCM-LM Frame Tx

The VOE Tx lookup for a valid CCM-LM PDU includes the following actions:

- Updating CCM-LM.TxFCf with Tx LM counter.
- Updating CCM-LM.TxFCb with the value of the VOP:VOE_CCM_LM:CCM_TX_FCB_CFG.CCM_TX_FCB register.This register is written by the VOE during CCM-LM Rx.
- Updating the CCM-LM.RxFCb with the value of the VOP:VOE_CCM_LM:CCM_RX_FCB_CFG.CCM_RX_FCB register. This register is written by the VOE during CCM-LM Rx.
- Counting the number of Tx CCM-LM frames (VOP:VOE_STAT:LM_PDU_CNT.LMM_TX_PDU_CNT).

Furthermore, RDI and sequence number are updated. For more information, see Continuity Check Messages (CCM), page 246.

3.24.7.5 CCM-LM Frame Rx

The VOE Rx lookup for a valid CCM-LM PDU includes the following actions.

- Updating the CCM_LM Rx sticky bit (VOP:VOE_STAT:OAM_RX_STICKY.CCM_LM_RX_STICKY).
- Storing the value of the CCM-LM.TxFCf (VOP:VOE_CCM_LM:CCM_TX_FCB_CFG.CCM_TX_FCB). This value is inserted into the CCM-LM PDU in the Tx direction. For more information, see CCM-LM Frame Tx, page 253.
- Sampling and storing the Rx LM counter, denoted RxFCf, (VOP:VOE_CCM_LM:CCM_RX_FCB_CFG.CCM_RX_FCB). This value is used to insert into the CCM-LM PDU in the Tx direction.

- There is no field dedicated to hold the value of the RxFCf in the CCM-LM PDU. However, there are 32 reserved bits following the CCM-LM.TxFCb, which can be used to hold the RxFCf value. The VOE can be configured to update the CCM-LM.reserved_field with value of RxFCf using the VOP::VOP_CTRL.CCM_LM_UPD_RSV_ENA bit field.
  If this option is not enabled, there is no way for the CPU to extract the value of RxFCf.
- Count the number of Rx CCM-LM frames in VOP:VOE_STAT:LM_PDU_CNT.LMR_RX_PDU_CNT.

3.24.8 Synthetic Loss Measurement

Synthetic loss measurement is implemented based on Y.1731 (03/2013). This includes a preliminary version of 1SL.

SynLM counts frames differently than F-LM. In particular, the LM counters are not incremented based on frame priority. When a VOE is enabled for SynLM, the LM counters are incremented based on the SynLM peer index. For more information about F-LM, see Frame Loss Measurement (F-LM), page 231.

3.24.8.1 Known Limitations

The VOEs supports SynLM for a single (programmable) priority towards 1 to 8 Peer MEPs. If more priorities must be supported, one VOE per additional priority must be added to implement the MEP.

The SLM/1SL is assumed to be transmitted using multicast address in the initiator MEP. There is no support for per peer Tx counting.

The implementation of SynLM reuses VOE configuration/statistics resources used for Frame Loss Measurement (LMM/LMR). This implies that a single VOE does not support SynLM simultaneously with the processing of Frame Loss Measurement PDUs.

If LMM/LMR PDUs are received on a VOE configured for SynLM they assert the correct Rx sticky bit. The rest of the functionality is not guaranteed. The same applies if SynLM PDUs are received on a VOE enabled for F-LM.

The SynLM support has the following known limitations in:

- VOEs enabled for SynLM support only one configurable priority.
- VOEs enabled for SynLM support only a single configurable SynLM Test ID (Initiator Function). The SynLM TEST ID is not verified in the Remote MEP.
- Only a single counter pr. VOE is supported on Initiator MEP Tx (SLM/1SL).
- SynLM counters support up to eight peer MEPs for Rx SLM, Tx SLR, and Rx SLR/1SL.
- Y.1731 requires that SLR PDUs received more than five seconds after the corresponding SLM is sent be discarded. This is *not* guaranteed by the VOE.
- Identification of the peer MEP is done solely based on the received MEPID received in the SynLM PDUs. There is no check of the SMAC or other.

3.24.8.2 SynLM Functional Overview

When configured for SynLM, the VOE must be programmed with a list of one to eight known SynLM peer MEPIDs. This list is denoted SynLM Peer List. Each MEPID identifies a peer MEP which is part of the SynLM session. I.e. Peer MEPs which either sends SLM PDUs to the current VOE or which responds to SLM sent from this VOE.

When a SynLM PDU is processed by the VOE, the SynLM Peer is identified by matching the Remote Peer MEPID (found in the SynLM PDU) against the SynLM Peer List. If the MEPID of the Remote MEP is found in the list, the Peer is assigned a SynLM Peer Index (sl_peer_idx) given by the row in the SynLM Peer List which matches the Remote MEP MEPID. If the Rx MEPID is not found in the list, the frame is marked as invalid.

SynLM LM Rx/Tx counters are updated based on the sl_peer_idx (not by frame priority). The decision whether to count the frame or not based on color is unchanged from F-LM.

A SynLM PDU mapped to a VOE, which is not enabled for SynLM (SLM/SLR/1SL), is not counted by the F-LM counters. However, it is counted as data in any Server Layer- or port VOEs. In reality, this should not happen because processing of SynLM and LMM/LMR are mutually exclusive.

SynLM as specified in Y.1731 (03/2013) includes the following two types of SynLM:

- Single-ended synthetic LM (SE-SynLM: SLM/SLR)
- Dual-ended synthetic LM (DE-SynLM: 1SL)

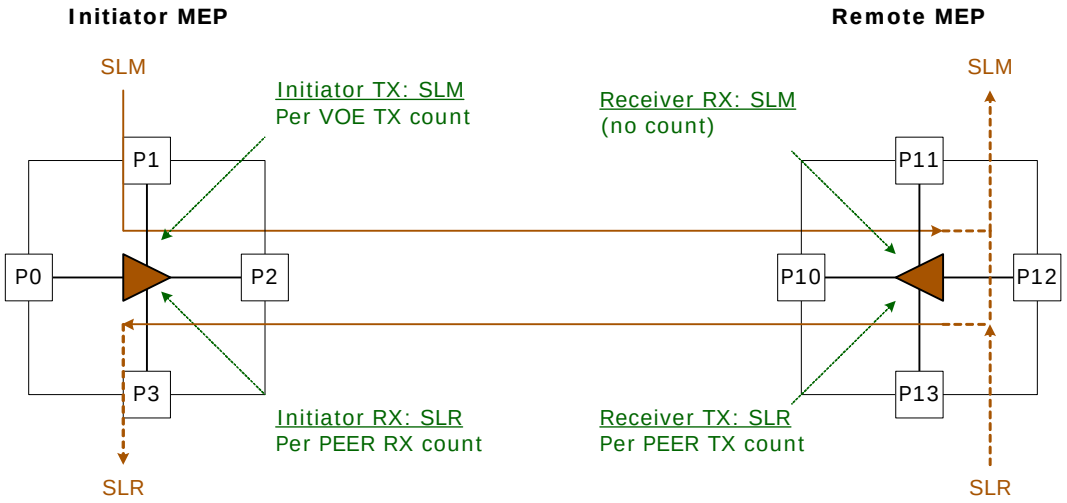
1SL processing shares resources with SLM/SLR processing, so running SE-SynLM and DE-SynLM are mutually exclusive.

Note that the type of SynLM running on a given connection depends only on the PDUs transmitted as part of the SynLM session, there is no specific VOE setting to configure single-ended versus dual-ended SynLM.

3.24.8.3 Single-Ended SynLM

The frame flow when running single-ended SynLM is shown in the following illustration. The illustration is valid both for Down- and Up-MEPs.

Figure 74 • **Single-Ended SynLM**



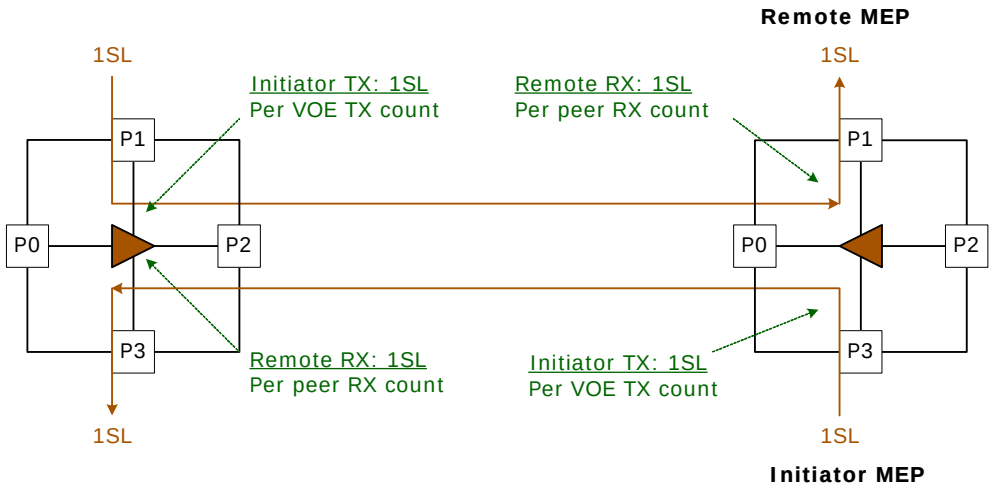
SLR PDUs transmitted from the Remote MEP can be either SLM PDUs looped by the VOE or SLR PDUs injected by the CPU. The Initiator MEP counts the number of Tx SLM PDUs. There is a dedicated counter for this purpose. The Remote MEP loops the SLM to SLR PDU, but no counters are implemented to count the incoming Rx SLM frames. The standard assumes that the Rx SLM count is identical to the Tx SLR count mentioned in the following:

- Remote MEP counts the number of Tx SLR frames per peer Initiator MEP, using Tx LM counters.
- Initiator MEP counts the number of Rx SLR frames per peer Remote MEP, using Rx LM counters.

3.24.8.4 SynLM: Dual-Ended LM

The frame flow when running dual-ended SynLM is shown in the following illustration. Dual-ended SynLM can optionally be bi-directional.

Figure 75 • **Dual-Ended SynLM**



The following counters are supported for dual-ended SynLM in the VOE:

- Initiator MEP counts the number of 1SL Tx PDUs per VOE. There is a dedicated counter for this.
- The Remote MEP counts the number of 1SL Rx PDUs per Initiator Peer MEP, using Rx LM counters. When running dual-ended SynLM, the Tx LM counters are not used.

3.24.8.5 SynLM vs. F-LM in the VOE

The implementation of SynLM affects the way F-LM using LMM/LMR and CCM-LM is done because of the way hierarchical counters are implemented in the VOP.

Only the active VOE processes the frame for each lookup to the VOP. The active VOE updates the LM counts of the port VOE and the server layer VOE (optional). Further, it is assumed that all frames mapped to the service VOE must be counted in the LM counters of the server layer VOE and the port VOE. This no longer is true when one or more of the VOEs in the hierarchy are enabled for SynLM.

A VOE configured for SynLM, only counts the number of SLM frames, which it processes. For more information, see Single-Ended SynLM, page 255 and Figure 75, page 256. No other frames are counted. As a result, when a service VOE updates the LM counters of the Server Layer VOE and/or the port VOE, it must first check whether either of the two are enabled for SynLM, in which case it does not update the LM counters of the server layer/port VOE.

Another implication is that for VOEs configured for SynLM, the LM counters are not incremented based on frame priority. They are updated based on the sl_peer_idx determined from the MEPID of the remote SynLM peer. For more information, see SynLM Functional Overview, page 254.

3.24.8.6 SynLM Configuration

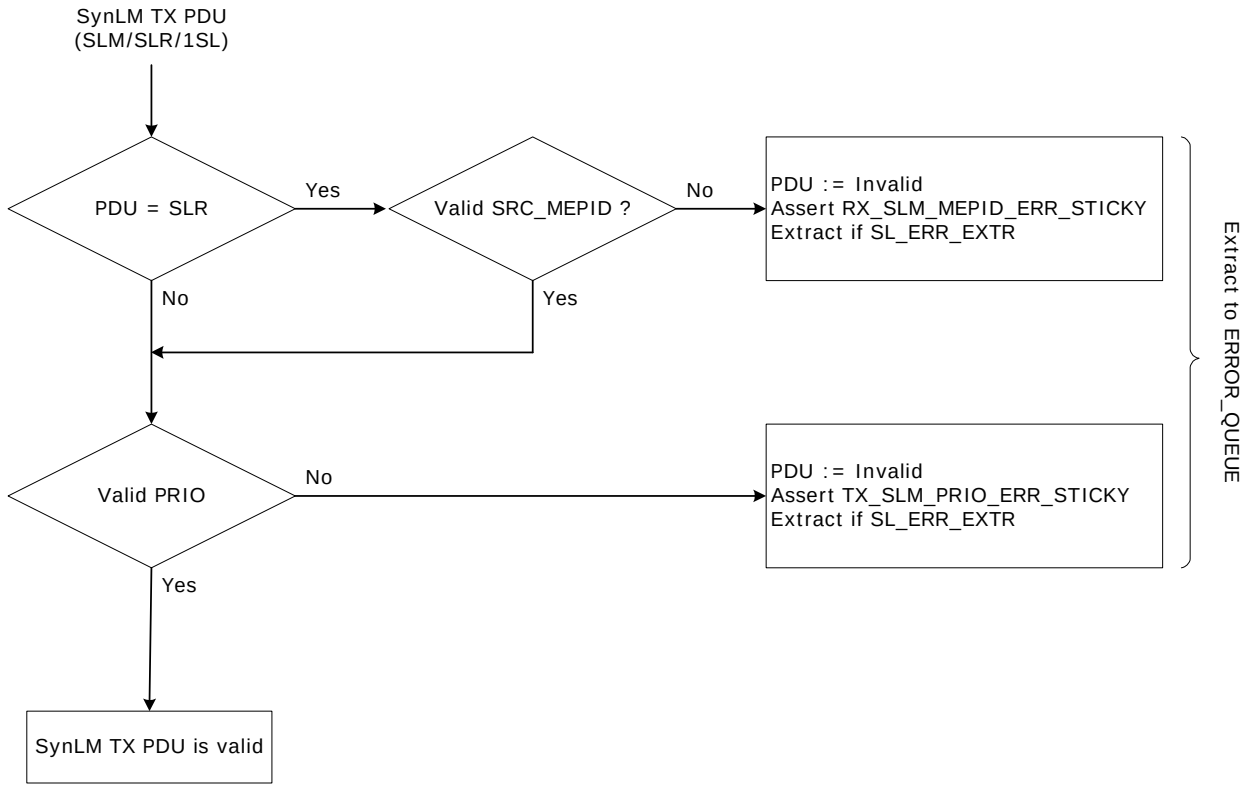
SynLM is configured using the following configuration:

- Enabling the VOE for SynLM (VOP:VOE_CONF_REG:VOE_MISC_CONFIG.SL_ENA).
- SynLM Peer List with eight peer MEPs (VOP:VOE_CONF:SLM_PEER_LIST(x).SLM_PEER_ENA and VOP:VOE_CONF:SLM_PEER_LIST(x).SLM_PEER_MEPID).
- Initiator MEP Test ID (VOP:VOE_CONF:SLM_TEST_ID.SLM_TEST_ID).
- SynLM Initiator MEPs own MEPID (VOP:VOE_CONF:VOE_MEPID_CFG.VOE_MEPID).
- SynLM priority (COSID) for the SynLM session (VOP:VOE_CONF:SLM_CONFIG.SLM_PRIO).

3.24.8.7 SynLM PDU Tx Validation

After the PDU validation of OAM PDUs, the SynLM PDUs are further validated before they are considered valid SynLM frames. The Tx validation performed for SynLM PDUs is shown in the following illustration.

Figure 76 • **SynLM Tx Validation**



If the Tx PDU is SLR, the VOE validates the SLR.SRC_MEPID. The SLR.SRC_MEPID must be found in the SynLM Peer List (VOP:VOE_CONF:SLM_PEER_LIST.SLM_PEER_MEPID).

For all SynLM Tx PDUs the frame priority is verified. The SLM/SLR/1SL priority must match the configured SynLM priority (VOP:VOE_CONF:SLM_CONFIG.SLM_PRIO).

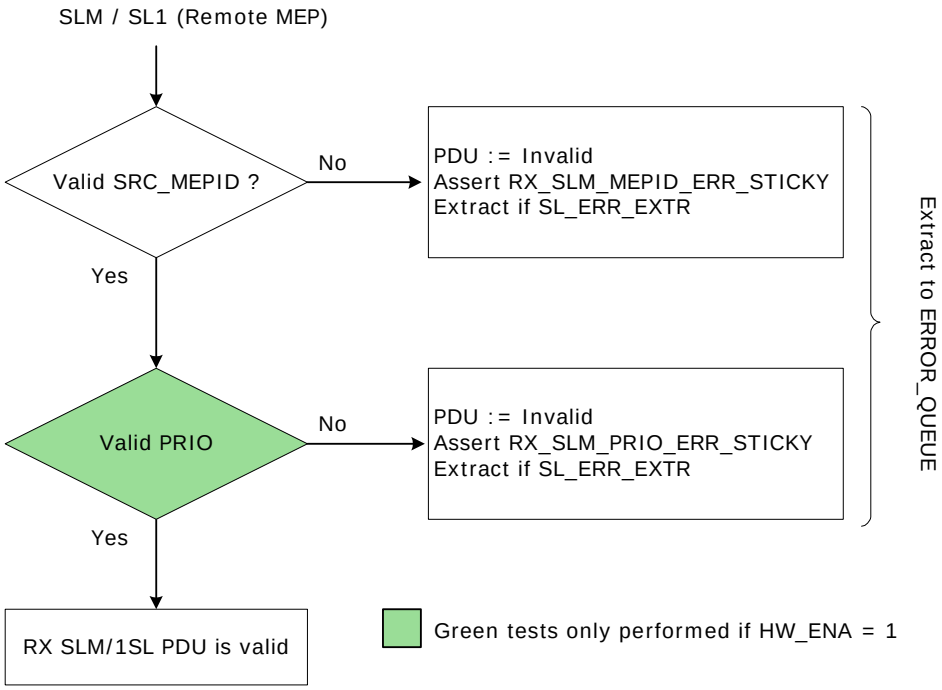
Sticky bits are asserted as illustrated.

Frames that fail the testing can optionally be extracted to the CPU error queue through VOP:VOE_STAT:PDU_EXTRACT.SL_ERR_EXTR.

3.24.8.8 SynLM PDU Rx Validation

The Rx validation performed for SLM/1SL PDUs is shown in the following illustration.

Figure 77 • **SLM/SL1 Rx Validation**



SLM/1SL Rx validation is as follows:

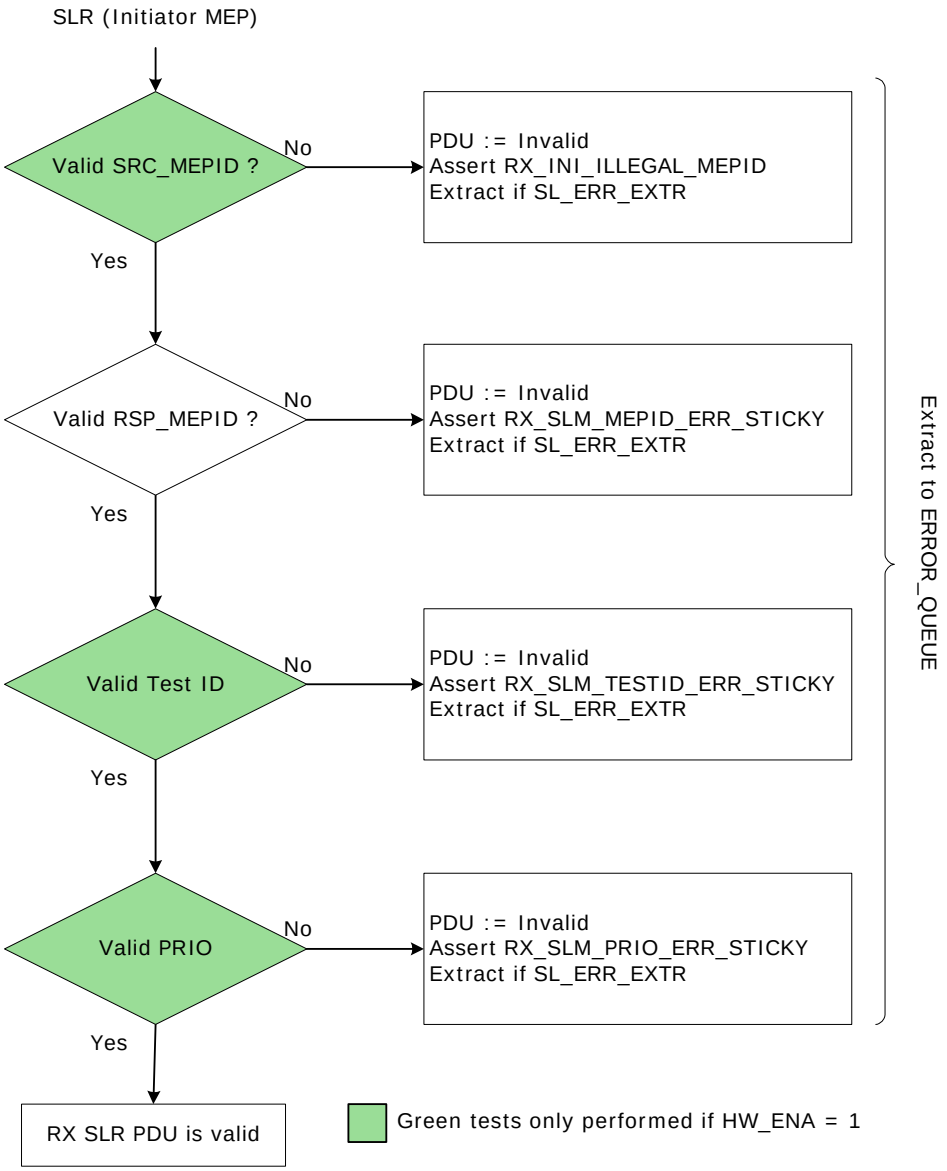
- SLM/1SL.SRC_MEPID must be found in the SynLM Peer List (VOP:VOE_CONF:SLM_PEER_LIST.SLM_PEER_MEPID).
- SLM/1SL priority must match the configured SynLM priority (VOP:VOE_CONF:SLM_CONFIG.SLM_PRIO).

Sticky bits are asserted as shown.

Frames that fail the testing can optionally be extracted to the CPU error queue through VOP:VOE_STAT:PDU_EXTRACT.SL_ERR_EXTR.

The Rx validation performed for SLR PDUs is shown in the following illustration.

Figure 78 • **SLR Rx Validation**



The following checks are performed:

- SLR.SRC_MEPID must match the MEPID configured for the SynLM session (VOP:VOE_CONF:VOE_MEPID_CFG.VOE_MEPID).
- SLR.RSP_MEPID must be found in the SynLM Peer List (VOP:VOE_CONF:SLM_PEER_LIST.SLM_PEER_MEPID).
- SLR.TEST_ID must match the TEST ID configured for the SynLM session (VOP:VOE_CONF:SLM_TEST_ID.SLM_TEST_ID).
- SLR priority must match the configured SynLM priority (VOP:VOE_CONF:SLM_CONFIG.SLM_PRIO).

Sticky bits are asserted as shown.

Frames that fail the testing can optionally be extracted to the CPU error queue through VOP:VOE_STAT:PDU_EXTRACT.SL_ERR_EXTR.

3.24.8.9 Enabling SLM/SLR/1SL Hardware Processing

The processing of SLM PDUs is enabled in VOP:VOE_CONF:OAM_HW_CTRL.LMM_ENA and SLR/1SL PDUs in VOP:VOE_CONF:OAM_HW_CTRL.LMR_ENA.

If the SynLM PDUs are not enabled for hardware processing, the SynLM PDUs are not modified. Statistics are updated correctly. Also the validation of Rx SynLM PDUs is modified. For more information, see SynLM PDU Tx Validation, page 256.

3.24.8.10 SLM/SLR/1SL Counter Configuration

The OAM PDU Tx/Rx counters are updated according to the counter configuration in SLM: VOP:VOE_CONF:OAM_CNT_OAM_CTRL.LMM_OAM_CNT_ENA and SLR/1SL: VOP:VOE_CONF:OAM_CNT_OAM_CTRL.LMR_OAM_CNT_ENA.

Valid SLM/SLR PDUs are always included in the F-LM counters. For more information, Frame Loss Measurement (F-LM), page 231.

3.24.9 Synthetic Loss Measurement, Single-Ended

Single-ended SynLM uses SLM/SLR PDUs to exchange loss measurement information between two MEPs.

3.24.9.1 Tx SLM

The VOE Tx lookup for a valid SLM PDU updates the SLM.TX_FCf with the SLM Tx frame count from VOP:VOE_STAT:SLM_TX_FRM_CNT.SLM_TX_FRM_CNT. After the update, the register value is incremented + 1 by the VOE.

3.24.9.2 Rx SLM

The VOE Rx lookup for a valid SLM PDU includes the following actions:

- Asserting the Rx sticky bit (VOP:VOE_STAT:OAM_RX_STICKY.SLM_RX_STICKY).
- Rx SLM PDUs are counted as selected/non-selected OAM (VOP:VOE_CONF:OAM_CNT_OAM_CTRL.LMM_OAM_CNT_ENA, shared with LMM).
- The VOE optionally loops valid SLM PDUs (VOP:VOE_CONF:LOOPBACK_ENA.LB_LMM_ENA, shared with LMM). If loopback is enabled, an SLR PDU is injected by the VOE in the Tx direction towards the peer MEP. For more information, see Looping OAM PDUs, page 245. The injected SLR is processed by VOE Tx lookup. For more information, see Tx SLR, page 260.
- The VOE optionally extracts all valid Rx SLM PDUs (VOP:VOE_CONF:OAM_CPU_COPY_CTRL.LMM_CPU_COPY_ENA, shared with LMM). When using this option, extraction is done regardless from which Peer the PDU is received.

3.24.9.3 Tx SLR

The VOE Tx lookup for a valid SLR PDU includes the following actions:

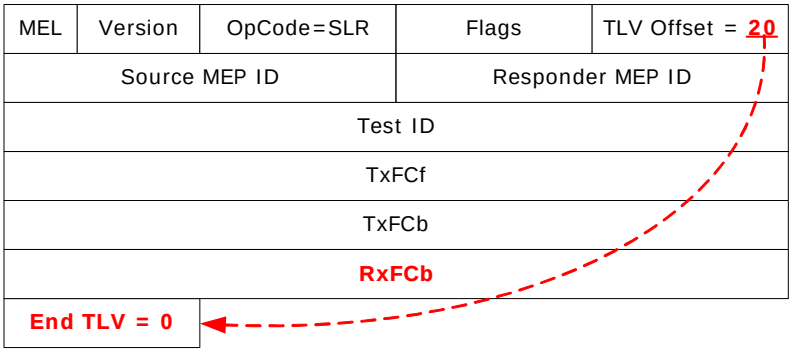
- Sampling the TxFCb value from the Tx LM counter selected by the SynLM peer index. TxFCb sample value is written to the SLR.TxFCb.
- Increasing the Rx LM counter. After sampling the TxFCb value selected by the SynLM_peer_idx.
- Setting Tx SLR.rsp_mepid to the locally configured MEPID value (VOP:VOE_CONF:VOE_MEPID_CFG.VOE_MEPID.

3.24.9.4 Rx SLR

The VOE Rx lookup for a valid SLR PDU asserts the VOP:VOE_STAT:OAM_RX_STICKY.SLR_RX_STICKY bit.

When the SLR is received at the Initiator MEP, the SynLM peer MEP Rx counter is sampled (RxFCb). However, there is no place in the SLR PDU for this value. In order to pass the RxFCb value to the CPU, the VOE optionally modifies the SLR frame to make space for the Rx counter. This update is enabled in VOP::VOP_CTRL.LMR_UPD_RXFCL_ENA. When the update is enabled, the VOE updates the SLR as shown in the following illustration.

Figure 79 • **SLR Updates**



The update includes the following actions:

- Updating the TLV Offset = 20 (points to the new End TLV).
- Inserting 32-bit RxFCb count right after the Tx FCb field. The value written into the SLR.RxFCb is the value of the Rx LM counter prior to processing the SLR PDU.
- Creating a new End TLV field (=0) immediately following the RxFCb count.
- Increasing the Rx LM counter after sampling the RxFCb value.

The Rx SLR can optionally be extracted to the CPU through VOP:VOE_STAT:SYNLM_EXTRACT.EXTRACT_PEER_RX. Extraction is optionally done as Hit-Me-Once with VOP:VOE_STAT:SYNLM_EXTRACT.EXTRACT_HMO.

The VOE optionally extracts all valid Rx SLR PDUs with VOP:VOE_CONF:OAM_CPU_COPY_CTRL.LMR_CPU_COPY_ENA. When using this option extraction is done, regardless of which Peer the PDU is received from.

3.24.10 Synthetic Loss Measurement, Dual-Ended

3.24.10.1 Tx 1SL

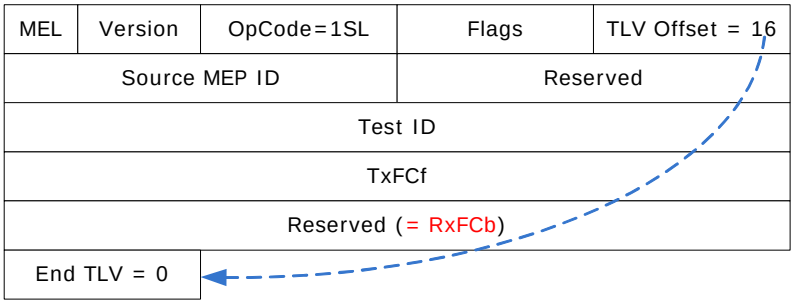
The VOE Tx lookup for a valid 1SL PDU updates the 1SL.TX_FCf with the 1SL Tx frame count from VOP:VOE_STAT:SLM_TX_FRM_CNT.SLM_TX_FRM_CNT. After the update, the register value is incremented + 1.

3.24.10.2 Rx 1SL

The VOE Rx lookup for a valid 1SL PDU includes the following actions:

- Asserting the VOP:VOE_STAT:OAM_RX_STICKY.SL1_RX_STICKY bit.
- When the 1SL PDU is received at the Remote MEP, the SynLM peer MEP Rx LM counter is sampled (RxFCb), selected by the SynLM peer index.
- Updating the Rx 1SL.Reserved with the RxFCb sample value. The 1SL PDU has a 32-bit reserved field right after TxFCf, used to carry the RxFCb value, shown in the following illustration.
- Increasing the Rx LM counter after sampling the RxFCb value.

Figure 80 • **Rx 1SL Updates**



The Rx 1SL can optionally be extracted to the CPU with VOP:VOE_STAT:SYNLM_EXTRACT.EXTRACT_PEER_RX. Extraction is optionally done hit-me-once with VOP:VOE_STAT:SYNLM_EXTRACT.EXTRACT_HMO.

All valid Rx 1SL PDUs can be extracted through VOP:VOE_CONF:OAM_CPU_COPY_CTRL.LMR_CPU_COPY_ENA.

3.24.11 Delay Measurement

The VOE supports two types of delay measurement specified by Y.1731:

- Single-ended DM (SE-DM): Use DMM and DMR PDUs.
- Dual-ended DM (DE-DM): Use 1DM PDUs.

VOE counters are shared between SE-DM and DE-DM, so a VOE only supports one type of DM at any given time. Note that the type of DM running on a given connection depends only on the PDUs transmitted as part of the DM session, there is no specific VOE setting to configure single ended versus dual ended DM. For more information about the two types of DM, see Single-Ended Delay Measurement (SE-DM: DMM/DMR), page 262 and section Dual-Ended Delay Measurement (DE-DM: 1DM), page 263.

The functionality is common for Up- and Down-VOEs, except for the precision of the time stamp. For more information, see DM Time Stamp Precision, page 262.

3.24.11.1 Time Stamping in External Device

A VOE can be configured to work with an external device, such as an external PHY. The external PHY does the time stamping of the DM PDUs, and the VOE performs the looping and extraction to CPU, but it does not alter the DM PDU TS fields.

This feature is configured commonly for all DM PDUs (1DM, DMM and DMR) in VOP:VOE_CONF:VOE_CTRL.EXTERN_DM_TSTAMP.

3.24.11.2 DM Time Stamp Precision

The time stamping in the Down-VOEs uses the same time stamping function as PTP. Hence the Down-MEP TS function is implemented with the same precision as PTP time stamping. For DM Time stamping in Up-VOEs the TS is sampled as follows:

- VOE Tx lookup: Tx TS is sampled when the frame passes OAM_PDU_MOD (ANA).
- VOE Rx lookup: Rx TS is sampled when the frame passes OAM_PDU_MOD (REW).

Because the reference point is the MAC at the residence port, this is not exactly as the time stamping used for PTP frames, but the time stamping is done on the correct side of the queuing system.

3.24.11.3 Selecting PTP Domain

The device implements three separate PTP time domains. In the VOP, each front port must be assigned to one of the three PTP time domains. DM PDUs received or transmitted on a given front port is time stamped in the PTP domain to which the port is assigned. PTP port configuration is done separately in OAM_PDU_MOD (ANA) and OAM_PDU_MOD (REW), but any front port must be assigned to the same PTP domain in both of the following places:

- OAM_PDU_MOD (ANA): ANA_AC_OAM_MOD:PDU_MOD_CFG:DM_PTP_DOMAIN_CFG.PTP_DOMAIN.
- OAM_PDU_MOD (REW): REW:PDU_MOD_CFG:DM_PTP_DOMAIN_CFG.PTP_DOMAIN.

3.24.12 Single-Ended Delay Measurement (SE-DM: DMM/DMR)

Single-ended DM (SE-DM) uses DMM/DMR PDUs to exchange timing information between two peer MEPs. The following description assumes internal time stamping (VOP:VOE_CONF:VOE_CTRL.EXTERN_DM_TSTAMP = 0). If external time stamping is used, the functionality is the same, except that none of the PDU time stamp fields are updated by the VOE.

3.24.12.1 Enabling DMM/DMR Hardware Processing

The processing of DMM PDUs is enabled in VOP:VOE_CONF:OAM_HW_CTRL.DMM_ENA and DMR PDUs in VOP:VOE_CONF:OAM_HW_CTRL.DMR_ENA. If DMM/DMR hardware processing is not enabled, only statistics are updated; no modifications are done to the DMM/DMR PDUs.

3.24.12.2 DMM/DMR Counter Configuration

The OAM PDU Tx/Rx counters are updated according to the counter configuration:

- DMM: VOP:VOE_CONF:OAM_CNT_OAM_CTRL.DMM_OAM_CNT_ENA.
- DMR: VOP:VOE_CONF:OAM_CNT_OAM_CTRL.DMR_OAM_CNT_ENA.

Valid DMM/DMR PDUs are optionally included in the F-LM counters:

- DMM: VOP:VOE_CONF:OAM_CNT_DATA_CTRL.DMM_DATA_CNT_ENA.
- DMR: VOP:VOE_CONF:OAM_CNT_DATA_CTRL.DMR_DATA_CNT_ENA.

3.24.12.3 DMM Frame Tx

The VOE Tx lookup for a valid DMM PDU includes the following actions:

- Updating the DMM.TxTSf field with Tx timing information.
- The VOE counts the number of valid Tx DMM PDUs (VOP:VOE_STAT:DM_PDU_CNT.DMM_TX_PDU_CNT).

3.24.12.4 DMM Frame Rx

The VOE Rx lookup for a valid DMM PDU includes the following actions:

- Updating DMM Rx sticky bit (VOP:VOE_STAT:OAM_RX_STICKY.DMM_RX_STICKY).
- Updating DMM.RxTSf with Rx timing information.
- The VOE optionally loops valid DMM PDUs (VOP:VOE_CONF:LOOPBACK_ENA.LB_DMM_ENA.) If loopback is enabled, an DMR PDU is injected by the VOE in the Tx direction towards the peer MEP. The injected DMR is processed by VOE Tx lookup. For more information, see Looping OAM PDUs, page 245 and DMR Frame Tx, page 263.
- The VOE counts the number of valid Rx DMM PDUs (VOP:VOE_STAT:DM_PDU_CNT.DMM_RX_PDU_CNT).
- The VOE optionally extracts all valid Rx DMM PDUs (VOP:VOE_CONF:OAM_CPU_COPY_CTRL.DMM_CPU_COPY_ENA).

3.24.12.5 DMR Frame Tx

The VOE Tx lookup for a valid DMR PDU includes the following actions:

- Updating DMR.TxTSb with Tx timing information.
- The VOE counts the number of valid Tx DMR PDUs (VOP:VOE_STAT:DM_PDU_CNT.DMR_TX_PDU_CNT).

3.24.12.6 DMR Frame Rx

The VOE Rx lookup for a valid DMR PDU includes the following actions:

- Updating the DMR Rx sticky bit (VOP:VOE_STAT:OAM_RX_STICKY.DMR_RX_STICKY).
- Updating DMR.RxTSb with Rx timing information.
- The VOE counts the number of valid Rx DMR PDUs (VOP:VOE_STAT:DM_PDU_CNT.DMR_RX_PDU_CNT).
- The VOE optionally extracts all valid Rx DMR PDUs (VOP:VOE_CONF:OAM_CPU_COPY_CTRL.DMR_CPU_COPY_ENA).

3.24.13 Dual-Ended Delay Measurement (DE-DM: 1DM)

Dual-ended delay measurement uses 1DM PDUs to exchange timing information between two peer MEPs. The following description assumes internal time stamping (VOP:VOE_CONF:VOE_CTRL.EXTERN_DM_TSTAMP = 0). If external time stamping is used, the functionality is the same, except that none of the PDU time stamp fields are updated by the VOE.

3.24.13.1 Enabling 1DM Hardware Processing

The processing of 1DM PDUs is enabled in VOP:VOE_CONF:OAM_HW_CTRL.SDM_ENA. If 1DM hardware processing is not enabled, only statistics are updated, no modifications are done to the 1DM PDU.

3.24.13.2 1DM Counter Configuration

The OM PDU Tx/Rx counters are updated according to the counter configuration in VOP:VOE_CONF:OAM_CNT_OAM_CTRL.SDM_OAM_CNT_ENA.

It is optional to include valid 1DM PDUs in the F-LM counters (VOP:VOE_CONF:OAM_CNT_DATA_CTRL.SDM_DATA_CNT_ENA). For more information, see Frame Loss Measurement (F-LM), page 231 through

3.24.13.3 1DM Frame Tx

The VOE Tx lookup for a valid 1DM PDU includes the following actions:

- Updating 1DM.TxTSf field with Tx time stamp.
- The VOE counts the number of valid Tx 1DM PDUs (VOP:VOE_STAT:DM_PDU_CNT.DMM_TX_PDU_CNT).

3.24.13.4 1DM Frame Rx

The VOE Rx lookup for a valid 1DM PDU includes the following actions:

- Updating 1DM Rx sticky bit (VOP:VOE_STAT:OAM_RX_STICKY.SDM_RX_STICKY).
- Updating 1DM.RxTSf with Rx time stamp.
- The VOE counts the number of valid Rx 1DM PDUs (VOP:VOE_STAT:DM_PDU_CNT.DMR_RX_PDU_CNT).
- The VOE optionally extracts all valid Rx 1DM PDUs (VOP:VOE_CONF:OAM_CPU_COPY_CTRL.SDM_CPU_COPY_ENA).

3.24.14 Generic/Unknown Opcodes

For information about classification of Generic- and Unknown OpCodes, see Generic PDU Support, page 237.

3.24.14.1 Enabling Generic/Unknown Hardware Processing

There is nothing to enable for Generic/Unknown OpCodes, because the VOE never updates these OAM PDUs.

3.24.14.2 Generic/Unknown OpCode Counter Configuration

The OAM PDU Tx/Rx counters are updated according to the counter configuration in VOP:VOE_CONF:OAM_CNT_OAM_CTRL.GENERIC_OAM_CNT_MASK and VOP:VOE_CONF:OAM_CNT_OAM_CTRL.UNK_OPCODE_OAM_CNT_ENA.

It is optional if Generic/Unknown PDUs are included in the F-LM counters through VOP:VOE_CONF:OAM_CNT_DATA_CTRL.GENERIC_DATA_CNT_MASK and VOP:VOE_CONF:OAM_CNT_DATA_CTRL.UNK_OPCODE_DATA_CNT_ENA.

3.24.14.3 VOE Pass Enable

Default behavior is to perform MEL filtering in the Rx/Tx direction; that is, to block OAM PDUs at the same MEL as the VOE. The VOE can be configured to let selected Generic OAM PDUs at the same MEL as the VOE pass rather be terminated due to MEL filtering through VOP:VOE_CONF:PDU_VOE_PASS.GENERIC_VOE_PASS_ENA. For more information about validating generic PDUs, see OAM PDU Validation: Rx Direction, page 240.

3.24.14.4 Generic/Unknown OpCode Tx

The VOE Tx lookup for Generic/Unknown PDUs does nothing.

3.24.14.5 Generic/Unknown OpCode Rx

The VOE Rx lookup for Generic/Unknown PDUs includes the following actions:

- Updating Rx sticky bit (VOP:VOE_STAT:OAM_RX_STICKY.GENERIC_RX_STICKY_MASK and VOP:VOE_STAT:OAM_RX_STICKY.UNK_OPCODE_RX_STICKY).

- The VOE optionally extracts all Generic/Unknown PDUs (VOP:VOE_CONF:OAM_CPU_COPY_CTRL.GENERIC_COPY_MASK and VOP:VOE_CONF:OAM_CPU_COPY_CTRL.UNK_OPCODE_CPU_COPY_ENA).
- The destination for Generic PDUs is configurable (VOP::OAM_GENERIC_CFG.GENERIC_OPCODE_CPU_QU).
- The destination for Unknownc PDUs is configurable (VOP::CPU_EXTR_CFG.DEF_COPY_QU).

3.24.15 Link Trace

The VOE support for LTM/LTR is limited to the support described for Generic OpCodes. LTM/LTR support can be viewed as a preconfigured Generic OpCode. The validation of LTM/LTR PDUs is limited to MEL and DMAC check.

3.24.15.1 LTM/LTR PDU Extraction

All valid LTM/LTR PDUs can be extracted to the CPU through VOP:VOE_CONF:OAM_CPU_COPY_CTRL.LTM_CPU_COPY_ENA and VOP:VOE_CONF:OAM_CPU_COPY_CTRL.LTR_CPU_COPY_ENA. LTM/LTR PDUs are extracted to the CPU queue in VOP:: CPU_EXTR_CFG_1.LT_CPU_QU.

3.24.15.2 LTM/LTR Counter Configuration

The OAM PDU Tx/Rx counters are updated according to the counter configuration:

- LTM (VOP:VOE_CONF:OAM_CNT_DATA_CTRL.LTM_DATA_CNT_ENA and VOP:VOE_CONF:OAM_CNT_OAM_CTRL.LTM_OAM_CNT_ENA)
- LTR (VOP:VOE_CONF:OAM_CNT_DATA_CTRL.LTR_DATA_CNT_ENA and VOP:VOE_CONF:OAM_CNT_OAM_CTRL.LTR_OAM_CNT_ENA)

3.24.15.3 LTM/LTR Rx Sticky Bit

When a valid LTM/LTR PDU is received, the VOP:VOE_STAT:OAM_RX_STICKY.LTM_RX_STICKY and VOP:VOE_STAT:OAM_RX_STICKY.LTR_RX_STICKY bit are asserted.

3.24.16 Non-OAM Sequence Numbering

For use with SAM testing (for example, RFC 2544) the VOE supports sequence number updating and checking of frames which are not Y.1731 OAM PDUs. The following frames can be used for non-OAM sequence numbering:

- Ethernet with ETYPE = 0x8880 and EPID = 0x0008. Other EtherTypes can be used through matching in VCAP CLM.
- UDP over IPv4/IPv6.

Within the scope of the VOP, non-OAM sequence numbering is denoted SAM_SEQ. Non-OAM frames with sequence number information are denoted SAM_SEQ frames. SAM_SEQ frames must be classified by the classifier to a special PDU type, used only for SAM_SEQ traffic: IFH.DST.ENCAP.PDU_TYPE = SAM_SEQ.

SAM_SEQ shares VOE resources with the OAM PDUs TST and LBM/LBR. Hence SAM_SEQ cannot be configured at the same time as any of these in the VOE.

The SAM_SEQ functionality is identical for VOEs configured for Up-VOE, Down-VOE and Port-VOE. SAM_SEQ is only supported for Ethernet VOEs. SAM_SEQ supports the use of per COSID sequence numbering. For more information, see SAM per COSID Sequence Numbering, page 246.

3.24.16.1 SAM_SEQ Functional Overview

SAM_SEQ frames contain a sequence number which can be checked at both ends of a connection. This allows for detection of out-of-sequence errors in the forward the and the return path between two MEPs.

SAM_SEQ frames are Ethernet or UDP frames over IPv4/IPv6. The proprietary sequence number format allows UDP frames to be updated in a way that does not affect the UDP checksum. The VOE does not validate that the UDP checksum is valid prior to the update. For more information about the sequence number format, see Non OAM Sequence Number Format, page 267.

Looping SAM_SEQ frames that are IPv4 or IPv6 requires modifying IP and UPD information in addition to the modifications done by the VOE. These modifications are configured in the analyzer.

In addition to enabling SAM_SEQ processing in the VOE, the SAM_SEQ frames must be classified to PDU_TYPE = SAM_SEQ through VCAP CLM action FWD_TYPE.

To support SAM_SEQ, the VOE is configured as either SAM_SEQ Initiator or SAM_SEQ Responder. These two functions are mutually exclusive, so a single VOE cannot support both functions at the same time.

A VOE enabled for SAM_SEQ internally classifies SAM_SEQ frames as one of the following PDU types:

- NON_OAM_MSG (Initiator Tx/Responder Rx)
- NON_OAM_RSP (Responder Tx/Initiator Rx)

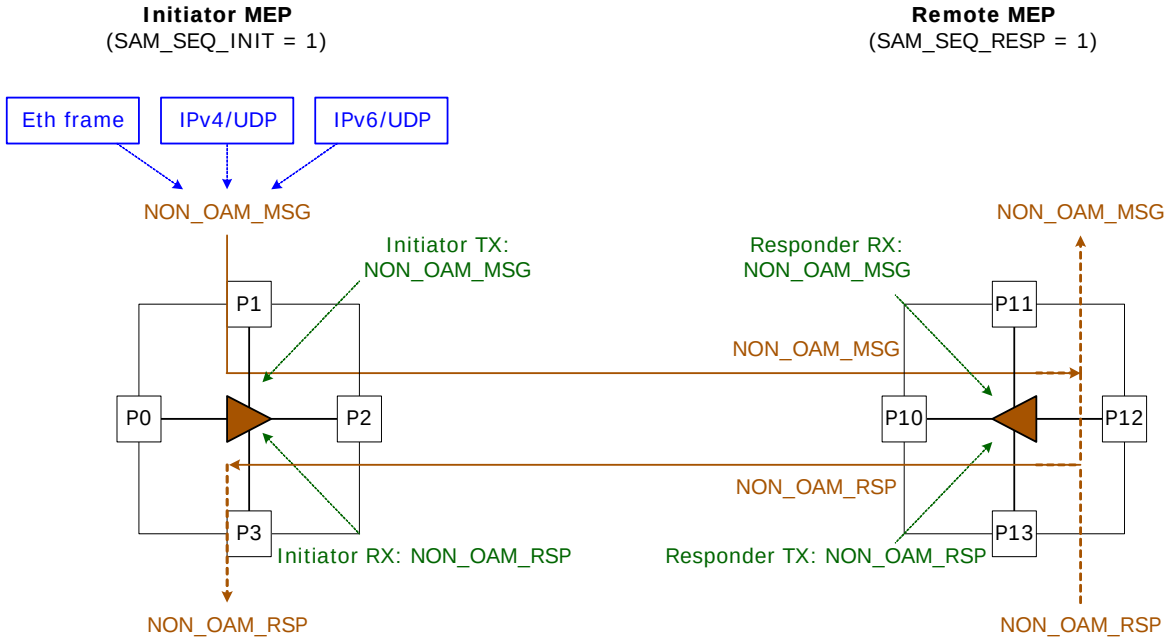
The classification to the above PDU types does not depend on whether the frames being processed are Ethernet, IPv4/UDP or IPv6/UDP. The classification as either NON_OAM_MSG or NON_OAM_RSP is determined only on VOE configuration (Initiator/Responder) and whether the frame is a Tx or an Rx frame. The classification as either NON_OAM_MSG or NON_OAM_RSP does not dependent on the content of the frame.

The VOE supports single-ended SAM_SEQ and dual-ended SAM_SEQs.

3.24.16.2 Single-Ended SAM_SEQ

A single-ended SAM_SEQ session consists of a bi-directional frame flow, where SAM_SEQ frames transmitted by the SAM_SEQ Initiator are looped at the SAM_SEQ Responder and subsequently received at the SAM_SEQ Initiator. Single-ended SAM_SEQ is shown in the following illustration. The illustration is valid for both Down- and Up-VOEs.

Figure 81 • **Single-Ended SAM_SEQ Frame Flow**



The SAM_SEQ Initiator transmits NON_OAM_MSG with sequence number PDUs towards the SAM_SEQ Responder.

The SAM_SEQ Responder checks the sequence number and in case of out-of-sequence errors, the SAM_SEQ Responder asserts FORWARD-SEQ-NUM-ERROR in the frame. At the SAM_SEQ Responder NON_OAM_MSGs are optionally looped and/or optionally extracted to CPU. NON_OAM_RSPs sent from the SAM_SEQ Responder are either NON_OAM_MSG being looped in the VOE or NON_OAM_RSP injected at the SAM_SEQ Responder.

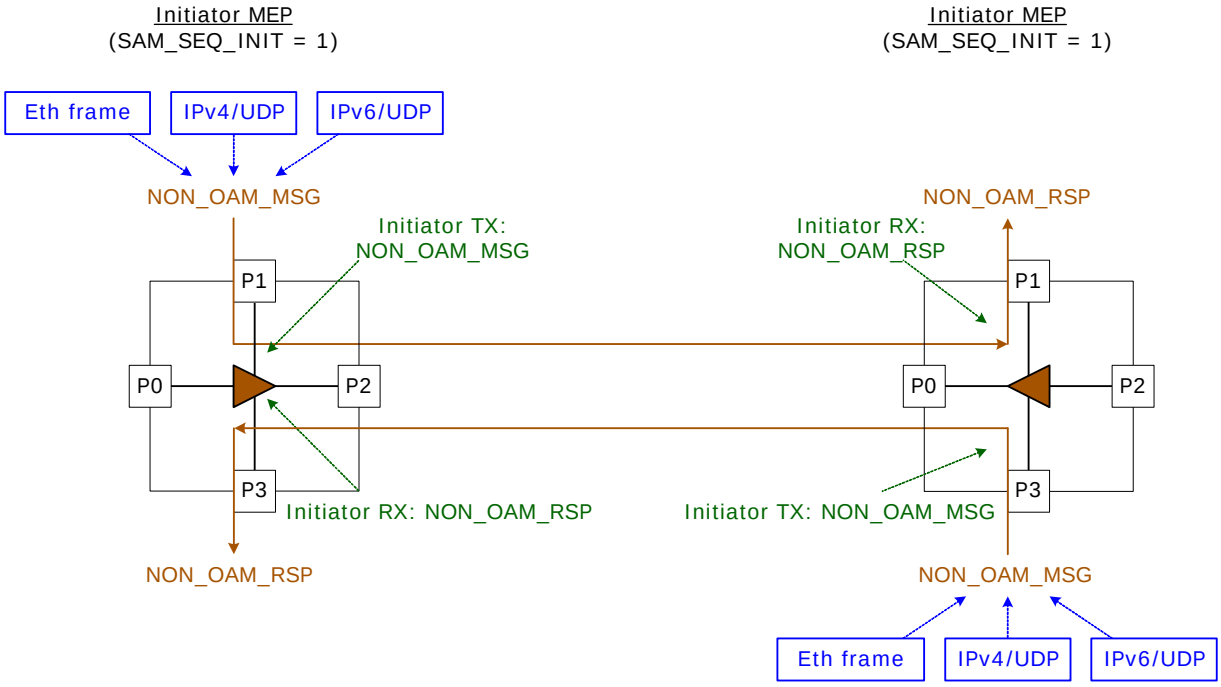
The SAM_SEQ Initiator validates the sequence number of Rx NON_OAM_RSP, updates the counters and stats. Subsequently the NON_OAM_RSP is optionally extracted to the CPU.

3.24.16.3 Dual-Ended SAM_SEQ

A dual-ended SAM_SEQ session consists of a uni-directional flow of SAM_SEQ frames. However, dual-ended SAM_SEQ can run in both directions (bi-directional dual-ended SAM), in which case frames flow in both directions.

Dual-ended SAM requires that the VOEs at both ends of the connection are configured as SAM_SEQ Initiators. Dual-ended SAM_SEQ (bi-directional) is shown in the following illustration. The illustration is valid for both Down- and Up-VOEs.

Figure 82 • **Dual-Ended SAM_SEQ Frame Flow**

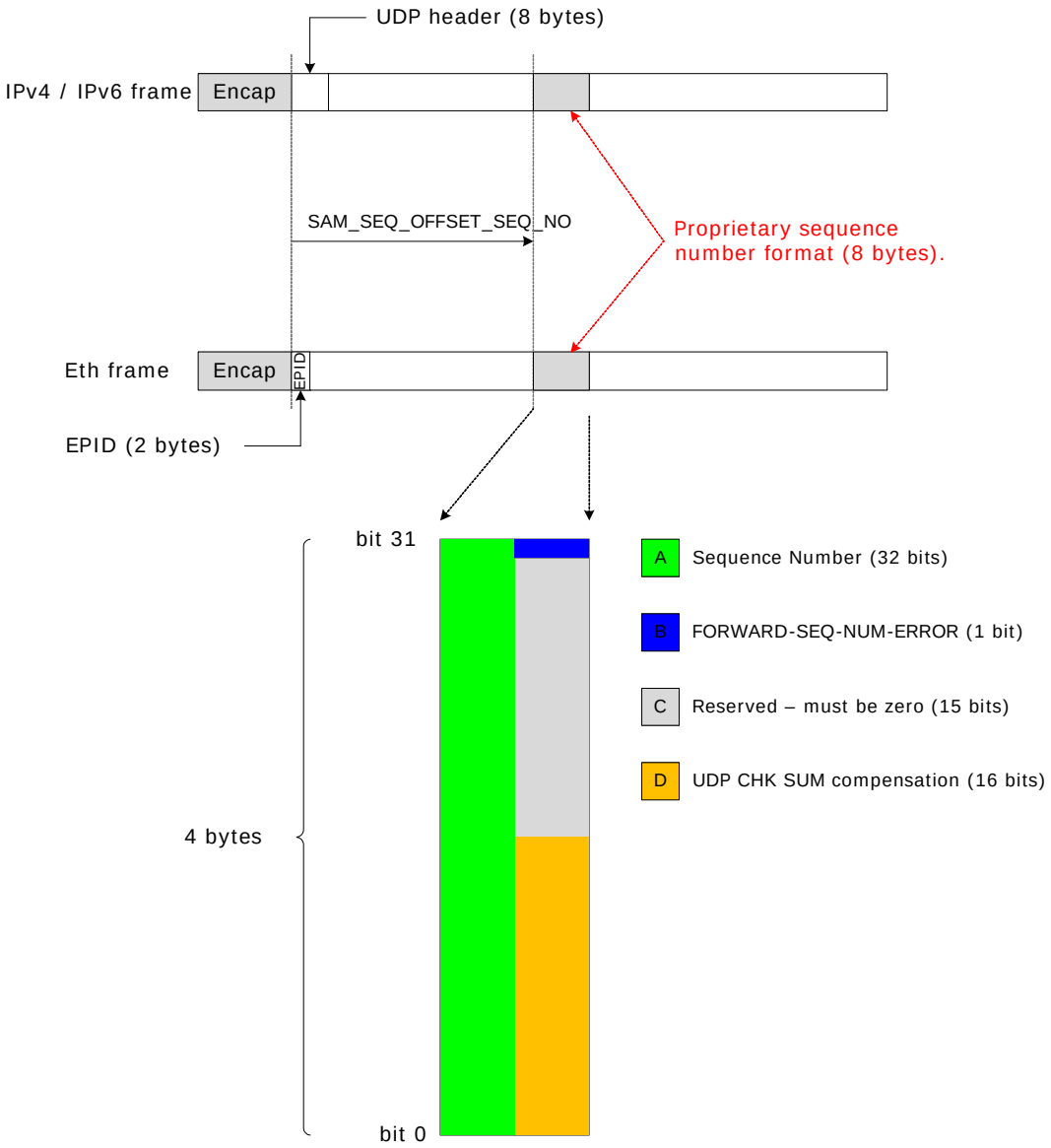


The SAM_SEQ initiator transmits NON_OAM_MSG with sequence number PDUs towards the peer MEP (also configured as SAM_SEQ Initiator). The peer MEP checks the sequence number of the Rx NON_OAM_RSPs and optionally extracts to the CPU.

3.24.16.4 Non OAM Sequence Number Format

The following illustration shows the proprietary sequence number format used for SAM_SEQ.

Figure 83 • **SAM_SEQ Number Format**



The location of the sequence number in the SAM_SEQ frames is configured as a 16-bit offset from the beginning of the SAM_SEQ PDU. Depending on the frame type the first byte of the SAM_SEQ PDU is defined as follows:

- Ethernet: First byte following ETYPE=0x8880, which is the EPID (=0x0008). For Ethernet frames this requires an offset of min 2 bytes, because the first 2 bytes of the Ethernet frame contain the EPID.
- IPv4/IPv6: First byte in the UDP frame. For UDP frames this requires an offset of min. 8 bytes, because the first 8 bytes of the UPD frame contain the UDP protocol information.

The SAM_SEQ format contains the following fields:

- Sequence number (32 bit): Sequence number inserted by the SAM_SEQ Initiator Tx lookup. Field must be initialized to zero when injecting the NON_OAM_MSG into the SAM_SEQ Initiator.
- FORWARD-SEQ-NUM-ERROR (1 bit): Indicates whether SAM_SEQ Responder Rx lookup detected a sequence number error in the received NON_OAM_MSG. Field must be initialized to zero when injecting the NON_OAM_MSG into the SAM_SEQ Initiator.
- Reserved (15 bit): Field must always be set to zero.

- UDP CHK SUM compensation (16 bit): Field is used for UDP checksum compensation, is updated when the SAM_SEQ is modified:
  - SAM_SEQ Initiator Tx lookup inserts sequence number
  - SAM_SEQ Responder Rx lookup asserts FORWARD-SEQ-NUM-ERROR.
    If SAM_SEQ is UDP over IPv4/IPv6 this field is updated in a way, so the UDP checksum is unaltered. In case of Ethernet SAM_SEQ frames this field is not used.

3.24.16.5 SAM_SEQ Validation

SAM_SEQ frames are not Y.1731 OAM PDUs, so there is no MEL filtering or version checking performed as part of the SAM_SEQ frame validation. Only the DMAC check is validated. Hence SAM_SEQ frames are considered valid when IFH.DST.ENCAP.PDU_TYPE = SAM_SEQ and the DMAC is valid.

In general, Tx OAM PDUs must be injected at the correct pipeline point to be processed correctly by the VOE Tx lookup. This is not the case for SAM_SEQ frames. If a SAM_SEQ frame is mapped to a VOE configured for SAM_SEQ in the Tx direction, the VOE processes the frame as a Tx SAM_SEQ frame, regardless of the injection pipeline point.

If SAM_SEQ frames are mapped to a VOE (Rx/Tx lookup) that is not enabled for SAM_SEQ, they are processed as data frames.

3.24.16.6 SAM_SEQ Common Configuration

This section describes the configuration common to both the SAM_SEQ initiator and responder.

The offset to the sequence number in the SAM_SEQ frames is configured in VOP:VOE_CONF:SAM_NON_OAM_SEQ_CFG.SAM_SEQ_OFFSET_SEQ_NO.

Whether the frame type used for SAM_SEQ requires updating the UDP checksum correction field is configured in VOP:VOE_CONF:SAM_NON_OAM_SEQ_CFG.SAM_SEQ_UPD_CHKSUM.

3.24.16.7 SAM_SEQ Initiator Configuration

The VOE is configured as a SAM_SEQ Initiator by setting VOP:VOE_CONF:SAM_NON_OAM_SEQ_CFG.SAM_SEQ_INIT to 1.

It is configurable which errors are counted upon receiving NON_OAM_RSP frames through VOP:VOE_CONF:SAM_NON_OAM_SEQ_CFG.SAM_SEQ_RX_ERR_CNT_ENA.

The VOE Tx lookup for a valid NON_OAM_MSG includes the following actions:

- Updating SAM_SEQ PDU Sequence Number with the contents of VOP:VOE_STAT:LBM_TX_TRANSID_CFG.LBM_TX_TRANSID. Subsequent to updating, increment the above field + 1.
- Optionally updating the UDP checksum correction field (VOP:VOE_CONF:SAM_NON_OAM_SEQ_CFG.SAM_SEQ_UPD_CHKSUM).

The VOE Rx lookup for a valid NON_OAM_RSP includes the following actions:

- Asserting the NON_OAM_MSG/NON_OAM_RSP sticky bit (VOP:VOE_STAT:OAM_RX_STICKY.NON_OAM_SEQ_RX_STICKY).
- Counting the number of Rx NON_OAM_RSP frames (VOP:VOE_STAT:LBR_RX_FRM_CNT.LBR_RX_FRM_CNT + 1).
- Comparing the sequence number of the frame to the previously Rx sequence number + 1 (VOP:VOE_STAT:LBR_RX_TRANSID_CFG.LBR_RX_TRANSID).
- The SAM_SEQ Initiator counts the number of out-of-sequence errors in the received NON_OAM_RSP frames depending on the configuration in VOP:VOE_CONF:SAM_NON_OAM_SEQ_CFG.SAM_SEQ_RX_ERR_CNT_ENA.
- The number of out-of-sequence errors are counted in the VOP:VOE_STAT:LBR_RX_TRANSID_ERR_CNT.LBR_RX_TRANSID_ERR_CNT counter.
- Asserting the VOP:VOE_STAT:OAM_RX_STICKY.LBR_TRANSID_ERR_STICKY bit when the above counter is incremented.
- Storing the sequence number of the current Rx NON_OAM_RSP frame (VOP:VOE_STAT:LBR_RX_TRANSID_CFG.LBR_RX_TRANSID).
- The NON_OAM_RSP frames containing out-of-sequence errors can optionally be extracted to the CPU (VOP:VOE_STAT:PDU_EXTRACT.SAM_RX_SEQ_ERR_EXTR).

The VOE optionally extracts the next valid NON_OAM_RSP PDU through VOP:VOE_STAT:PDU_EXTRACT.RX_TEST_FRM_NXT_EXTR. All NON_OAM_RSP frames received by the SAM_SEQ Initiator can optionally be copied to the CPU using the configuration in VOP:VOE_CONF:OAM_CPU_COPY_CTRL.LBR_CPU_COPY_ENA. Frames are extracted to the CPU queue in VOP::CPU_EXTR_CFG_1.LBR_CPU_QU.

3.24.16.8 SAM_SEQ Responder Configuration

The VOE is configured as a SAM_SEQ Responder by setting VOP:VOE_CONF:SAM_NON_OAM_SEQ_CFG.SAM_SEQ_RESP to 1.

The VOE Rx lookup for a valid NON_OAM_MSG includes the following actions:

- Asserting the NON_OAM_MSG/NON_OAM_RSP sticky bit (VOP:VOE_STAT:OAM_RX_STICKY.NON_OAM_SEQ_RX_STICKY).
- Counting the number of Rx NON_OAM_MSG frames (VOP:VOE_STAT:LBR_RX_FRM_CNT.LBR_RX_FRM_CNT).
- Comparing the sequence number of the incoming frame to the sequence number of the previous frame + 1 (VOP:VOE_STAT:LBR_RX_TRANSID_CFG.LBR_RX_TRANSID).
- If the sequence number of the incoming frame does not match expectations, the VOE asserts the FORWARD-SEQ-NUM-ERROR bit in the incoming frame. The VOE further optionally updates the UDP checksum correction field, depending on the configuration: VOP:VOE_CONF:SAM_NON_OAM_SEQ_CFG.SAM_SEQ_UPD_CHKSUM
- Counting the number of out-of-order Rx NON_OAM_MSG frames (VOP:VOE_STAT:LBR_RX_TRANSID_ERR_CNT.LBR_RX_TRANSID_ERR_CNT).
- If the sequence number of the incoming frame does not match expectation, the VOP:VOE_STAT:OAM_RX_STICKY.LBR_TRANSID_ERR_STICKY sticky bit is asserted.
- The NON_OAM_RSP frames containing out-of-sequence errors are optionally extracted (VOP:VOE_STAT:PDU_EXTRACT.SAM_RX_SEQ_ERR_EXTR).
- SAM_SEQ Responder optionally loops the NON_OAM_MSG (VOP:VOE_CONF:LOOPBACK_ENA.LB_LBM_ENA). If loopback is enabled, an NON_MSG_RSP is injected by the VOE in the Tx direction towards the peer MEP. For more information, see Looping OAM PDUs, page 245.

The VOE optionally extracts all NON_OAM_MSG frames through VOP:VOE_CONF:OAM_CPU_COPY_CTRL.LBR_CPU_COPY_ENA. Frames are extracted to the CPU queue in VOP::CPU_EXTR_CFG_1.LBR_CPU_QU.

The VOE Tx lookup for a valid NON_OAM_RSP includes the following actions:

- Counting the number of NON_OAM_RSP transmitted (VOP:VOE_STAT:LBR_TX_FRM_CNT.LBR_TX_FRM_CNT).
- No changes are made to the NON_OAM_RSP PDU when it is transmitted from the SAM_SEQ Responder.

3.24.17 Service Activation Test (SAT)

This section describes the required VOE configurations to support SAT.

When running SAT, the UNI-N is logically moved inside the device. This affects only the outer Up-VOE in the VOE hierarchy, consequently only the outer Up-VOE can be configured for SAT. The remaining VOEs in the VOE hierarchy are unaffected by SAT configuration. This includes all Down-VOEs, so the following description only includes Up-VOEs.

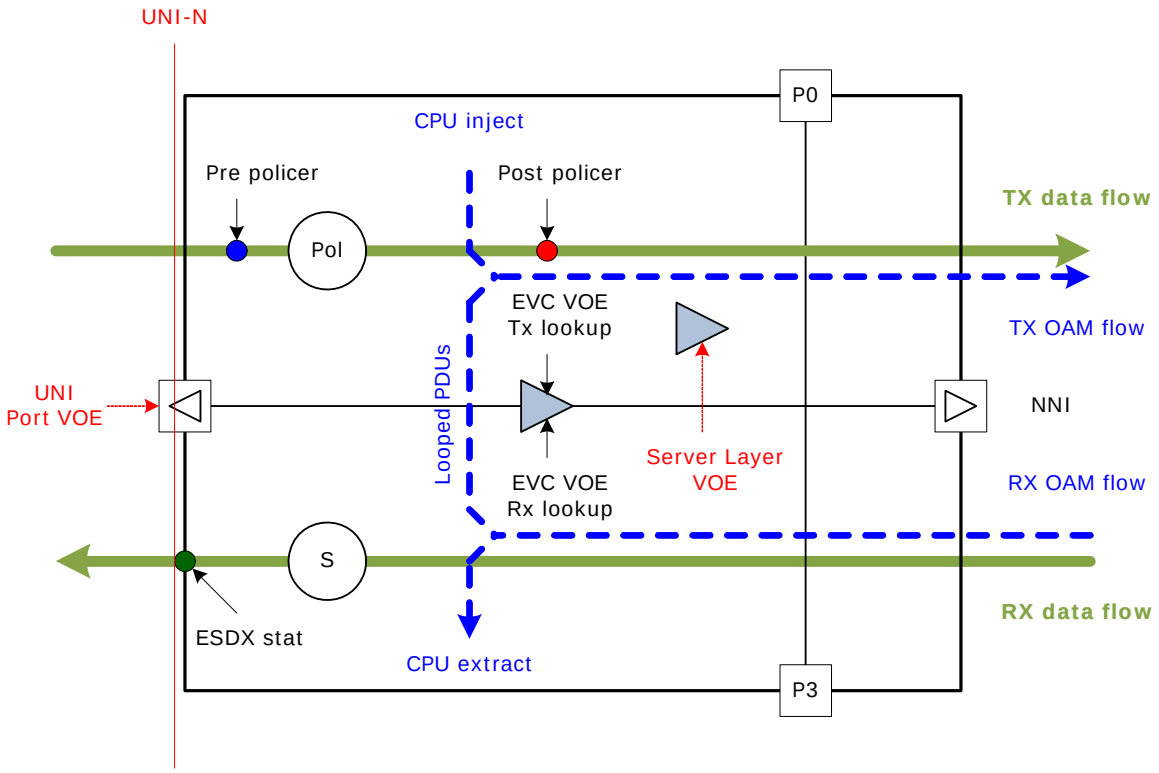
SAT configuration only affects counters inside and outside the VOP all other VOE functionality described in the previous is unaffected.

SAT is only supported for Ethernet VOE.

3.24.17.1 Standard VOE Setup

The following illustration shows the frame flow of a standard Up-VOE hierarchy. The outer Up-VOE is configured as an EVC VOE and the inner Up-VOE as a Server Layer VOE.

Figure 84 • **Standard VOE Hierarchy**



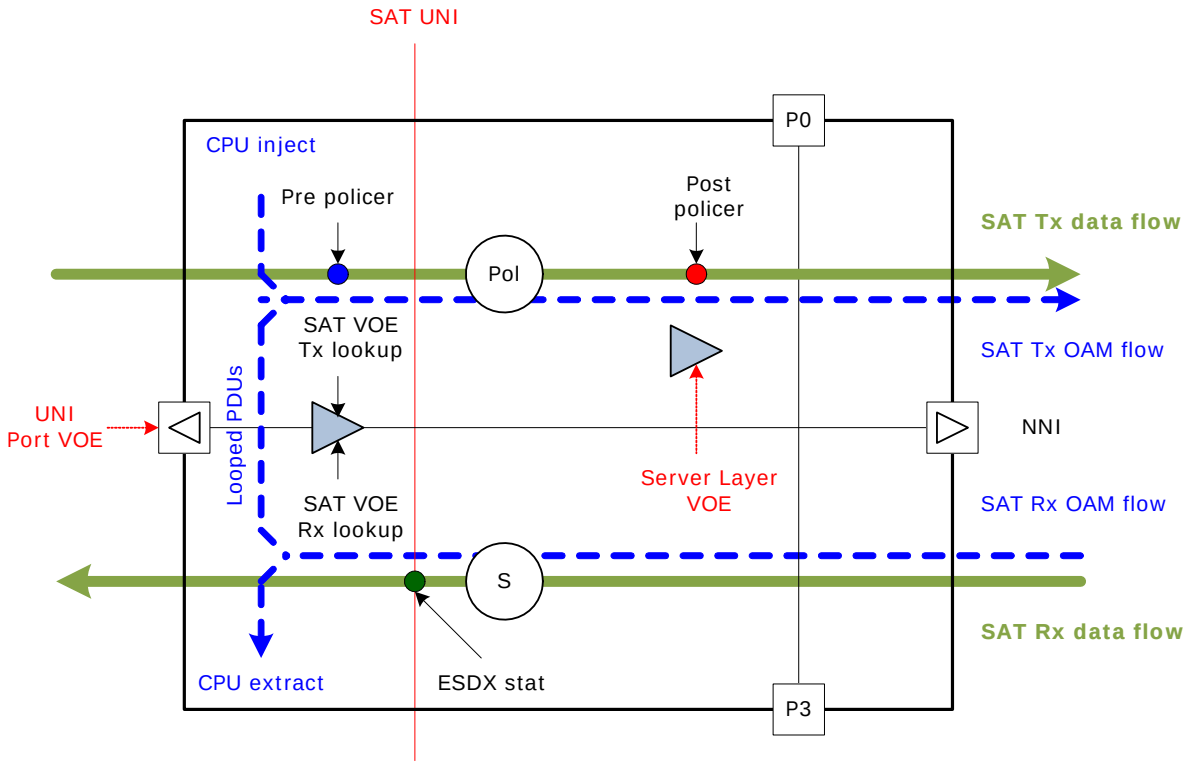
In the standard setup, the UNI-N is located at the same point as the UNI port VOE. The following are important details to understanding the VOE SAT configuration.

- **Ingress policer.** Tx data frames pass through the ingress Policer prior to the VOE Tx lookup. OAM PDUs are injected/looped into the ingress path after the Policer. I.e. for all frame types, the VOE Tx lookup processes frames after ingress policing. This implies that frames discarded by the policer are never processed by the outer Up-VOE. If data frames are re-colored by the policer, the VOE must use the frame color after the policer (post-policer color).
- **Egress shaper.** Egress Rx OAM PDUs are processed by the VOE Rx lookup prior to the shaper in the egress path. The VOE either terminates, extracts or loops all valid OAM PDUs. This implies that OAM PDUs processed by the VOE never pass through the egress shaper and consequently they are never counted by the egress shaper bucket.
- **ESDX counters.** Egress Rx OAM PDUs are processed by the VOE Rx lookup prior to the shaper in the egress path. The VOE either terminates, extracts or loops all valid OAM PDUs. This implies that OAM PDUs processed by the VOE are never transmitted on the UNI-N, and consequently, are never counted by the UNI ESDX counters.

3.24.17.2 SAT VOE Setup

A VOE configured as an outer Up-VOE can be configured as SAT VOE through VOP:VOE_CONF:VOE_CTRL.SAT_TEST_VOE. When a VOE is configured for SAT, the UNI is logically moved inside the switch (denoted SAT UNI). Further the outer Up-VOE (configured for SAT) is moved to a place between the UNI port VOE and the SAT UNI. The following illustration shows the location of the SAT UNI and the SAT VOE, when the VOE is enabled for SAT.

Figure 85 • **SAT VOE Hierarchy**



The important differences between the SAT VOE and the standard VOE are:

- **Ingress policer.** Tx data frames pass through the ingress Policer after the SAT VOE Tx lookup. OAM Tx PDUs are injected/looped into the ingress path before the Policer. I.e. for all frame types, the SAT VOE Tx lookup processes frames before ingress policing. This implies that frames discarded by the policer must be processed by the SAT Up-VOE (but not by the Server Layer VOE). If data frames are re-colored by the policer, the SAT VOE uses the frame color before the policer (pre-policer color), while the Server Layer VOE uses post-policer color for all frames including the SAT VOE OAM PDUs.
- **Egress shaper.** Egress Rx OAM PDUs are processed by the SAT VOE Rx lookup after the shaper in the egress path. This implies that OAM PDUs processed by the VOE pass through the egress shaper and consequently they MUST be counted by the egress shaper bucket.
- **ESDX counters.** Egress Rx OAM PDUs are processed by the SAT VOE Rx lookup after the shaper in the egress path. This implies that OAM PDUs processed by the VOE pass the SAT UNI and consequently they MUST be counted by the UNI ESDX counters.

3.24.18 G.8113.1 Specific Functions

This section describes special VOE functions for G.8113.1 OAM processing. These functions are enabled in VOP:VOE_CONF:VOE_CTRL.G_8113_1_ENA.

3.24.18.1 LBM/LBR TLV processing

When configured for G.8113.1, the VOE can verify the following TLVs specified by G.8113.1:

- Target MEP / MIP ID TLV (LBM, mandatory)
- Replying MEP / MIP ID TLV (LBR, mandatory)
- Requesting MEP ID TLV (LBM/LBR optional)

The VOE verifies the TLVs upon PDU reception and updates them when transmitting the PDUs. When an LBR frame is looped by the VOE, the following TLV modifications are included:

- Verifing the LBM “target MEP ID” TLV
- Removing “target MEP ID” TLV

- Inserting “replying MEP ID” TLV into the generated LBR frame
- Updating “requesting MEP” TLV (if present) with MEP ID info and set the loopback indicator = 1

When receiving LBR PDUs that include a “requesting MEP ID TLV”, the VOE can optionally verify that the value of the loopback indication (LBI) field is = 1 in VOP::VOP_CTRL.G_8113_1_LBK_INDC_CHK_ENA.

When verifying the MEP/MIP IDs located in the TLVs, the errors can optionally be counted in VOP::VOP_CTRL.G_8113_1_CNT_LBR_RX_ERROR_ENA.

If error counting is enabled, errors are counted in the following counters, shared with CCM(-LM) MEP ID error counting:

- VOP:VOE_STAT:CCM_RX_ERR_1.CCM_RX_MEPID_ERR_CNT
- VOP:VOE_STAT:CCM_RX_ERR_1.CCM_RX_MEGID_ERR_CNT

When a VOE is configured for G.8113.1, it can operate in different modes:

- Discovery mode
- Connection Verification for MIP
- Connection Verification for MEP

The VOE must be configured for the current operational mode (VOP:VOE_CONF:G_8113_1_CFG.G_8113_1_INITIATOR_FUNCTION).

When receiving LBM/LBR PDUs, it is optional if the MIP/MEP IDs contained in the TLVs are validated (VOP:VOE_CONF:G_8113_1_CFG.G_8113_1_LBX_MEXID_CHK_ENA).

If any of the TLV checks performed fail, the OAM PDU is marked as invalid and processed as an invalid LBM/LBR frame.

When MEP ID validation is enabled, the MEP IDs in the TLVs are verified against the values in the following registers:

- VOP:VOE_CONF:VOE_MEPID_CFG.VOE_MEPID (Local MEP ID)
- VOP:VOE_CONF:PEER_MEPID_CFG.PEER_MEPID (Remote MEP ID)

When MIPID validation is enabled, the remote MIPID in the TLVs is verified against the values in the following registers:

- VOP:VOE_CONF:G_8113_1_REMOTE_MIPID.G_8113_1_REMOTE_MIPID
- VOP:VOE_CONF:G_8113_1_REMOTE_MIPID1.G_8113_1_REMOTE_MIPID1
- VOP:VOE_CONF:G_8113_1_REMOTE_MIPID2.G_8113_1_REMOTE_MIPID2
- VOP:VOE_CONF:G_8113_1_REMOTE_MIPID3.G_8113_1_REMOTE_MIPID3

If the Rx validation of MEP-/MIPID fails, errors are reported in the following sticky bits:

- VOP:VOE_STAT:OAM_RX_STICKY2.G_8113_1_LBX_RX_MISSING_TLV_STICKY
- VOP:VOE_STAT:OAM_RX_STICKY2.G_8113_1_LBX_RX_ILLEGAL_SUBTYPE_STICKY
- VOP:VOE_STAT:OAM_RX_STICKY2.G_8113_1_LBX_RX_ILLEGAL_MEXID_STICKY
- VOP:VOE_STAT:OAM_RX_STICKY2.G_8113_1_LBR_RX_ILL_LBK_IND_STICKY

OAM PDUs failing the Rx validation can optionally be extracted to the CPU by setting the following bits.

- VOP:VOE_STAT:PDU_EXTRACT.G_8113_1_LBM_RX_ERR_EXTR
- VOP:VOE_STAT:PDU_EXTRACT.G_8113_1_LBR_RX_ERR_EXTR

3.24.18.2 MEL Check Disable

For MPLS OAM, there is no need for a MEL as is the case for Y.1731 OAM. However, the G.8113.1 PDUs contain a MEL value.

If the VOE is enabled for G.8113.1 OAM, it is possible to disregard the MEL checking of the G.8113.1 PDU and blindly accept all MEL values. This is done at the VOP level in VOP:COMMON:VOP_CTRL.G_8113_1_MEL_CHK_DIS.

If MEL checking is enabled, G.8113.1 OAM PDUs will be MEL filtered in the same ways as Y.1731 OAM PDUs.

3.24.18.3 IFH.TTL Reset

When a VOE is configured for G.8113.1 OAM, the VOE resets the value of IFH.TTL = 1 when looping OAM PDUs. This is done to support looping PW OAM PDUs carried in VCCV3 associated channel.

3.25 VOE: MPLS-TP OAM

The VOE can be configured to support MPLS-TP OAM. MPLS-TP Bidirectional Forwarding Detection (BFD) as specified in RFC-6428 is supported by the VOE when configured for MPLS-TP operation.

3.25.1 MPLS-TP VOE Functions

In addition to the BFD protocol, the VOE supports configuration of eight Generic G-ACH Channel Types. For more information, see Generic G-ACH Channel Types, page 238.

MPLS-TP VOEs are configured in register target VOP_MPLS. This target is overlaid with the Ethernet VOE register target VOP. As a result, all registers in VOP_MPLS must be written their default values before use.

3.25.1.1 OAM MPLS-TP PDU Counters

Each PDU type can be configured as a selected PDU type or a non-selected PDU type through VOP_MPLS:VOE_CONF_MPLS:OAM_CNT_SEL_MPLS. When a valid PDU is processed by the VOE, it is counted as either a selected or non-selected PDU in the corresponding Rx/Tx counter.

Selected Tx PDUs are counted in VOP_MPLS:VOE_STAT_MPLS:TX_CNT_SEL_OAM_MPLS.TX_CNT_SEL_OAM_MPLS and non-selected Tx PDUs are counted in VOP_MPLS:VOE_STAT_MPLS:TX_CNT_NON_SEL_OAM_MPLS.TX_CNT_NON_SEL_OAM_MPLS.

Selected Rx PDUs are counted in VOP_MPLS:VOE_STAT_MPLS:RX_CNT_SEL_OAM_MPLS.RX_CNT_SEL_OAM_MPLS and non-selected Rx PDUs are counted in VOP_MPLS:VOE_STAT_MPLS:RX_CNT_NON_SEL_OAM_MPLS.RX_CNT_NON_SEL_OAM_MPLS.

3.25.1.2 Generic/Unknown G-ACH

The VOE supports configuring up to 8 G-ACH Channel Types as Generic G-ACH Channel Types. The Generic G-ACH Channel Types can be extracted to the CPU and the VOE updates dedicated statistics for these PDUs. The Generic G-ACH Channel Types are programmed in the VOP in VOP::MPLS_GENERIC_CODEPOINT.

The following configuration and statistics are implemented per VOE for Generic G-ACH Channel Types:

- A sticky bit is asserted when a PDU with a Generic G-ACH Channel Type is received (VOP_MPLS:VOE_STAT_MPLS:CPT_RX_STICKY_MPLS.GENERIC_CPT_RX_STICKY_MASK).
- Generic G-ACH Channel Type PDUs can be extracted to the CPU (VOP_MPLS:VOE_CONF_MPLS:CPU_COPY_CTRL_MPLS.GENERIC_COPY_MASK).

If a PDU is received with a G-ACH Channel Type that is not recognized as BFD or a Generic G-ACH Channel Type, it is classified as an Unknown G-ACH Channel Type.

The following configuration and statistics are implemented per VOE for Unknown G-ACH Channel Types:

- A sticky bit is asserted when a PDU with an Unknown G-ACH Channel Type is received (VOP_MPLS:VOE_STAT_MPLS:CPT_RX_STICKY_MPLS.UNK_CPT_RX_STICKY).
- PDUs with Unknown G-ACH Channel Type can be extracted to the default CPU queue (VOP_MPLS:VOE_CONF_MPLS:CPU_COPY_CTRL_MPLS.UNK_CPT_CPU_COPY_ENA).

3.25.1.3 Basic MPLS-TP VOE Setup

The VOE is enabled for MPLS-TP OAM by setting VOP:VOE_CONF_REG:VOE_MISC_CONFIG.MPLS_OAM_ENA to 1. When set, the VOE use the VOE_CONF_MPLS and VOE_STAT_MPLS register groups.

The VOE is enabled in VOP_MPLS:VOE_CONF_MPLS:VOE_CTRL_MPLS.VOE_ENA. When not enabled, the VOE can be configured, but no frame processing is done.

The VOE is either configured as an Up-MEP or a Down-MEP through VOP_MPLS:VOE_CONF_MPLS:VOE_CTRL_MPLS.UPMEP_ENA.

3.25.2 Bidirectional Forwarding Detection (BFD) Implementation

The bidirectional forwarding detection (BFD) implementation supported by the VOE is described in RFC-6428. It is assumed that the BFD protocol is implemented partly by the VOE and partly by a CPU. It is entirely up to the CPU to control the transmitting of BFD frames using either the AFI or manual CPU-based frame injection.

3.25.3 BFD Functional Overview

This section provides an overview of the BFD supported in the device.

3.25.3.1 BFD Limitations

The BFD implementation in the device has the following limitations:

- The VOE never modifies the local BFD state nor the BFD diagnostic code.
- The VOE cannot verify the Source MEP-ID TLV, which is part of the BFD CV frame.
- The VOE cannot verify the Authentication Section of a BFD PDU.

3.25.3.2 Valid BFD G-ACH Channel Types

The following G-ACH Channel Types are accepted as being BFD PDUs by the VOE:

- 0x7: BFD CC (RFC-5885)
- 0x22: BFD CC (RFC-6428)
- 0x23: BFD CV (RFC-6428)

The G-ACH Channel Type used by the VOE for detection of BFD CC PDUs is configurable in VOP_MPLS:VOE_CONF_MPLS:BFD_CONFIG.BFD_CC_RFC6428.

3.25.3.3 BFD Connection Types

A BFD connection is a point to point connection between two peer MPLS MEPs. As part of the BFD connection, the two MEPs exchange BFD Connection Verification (CV) PDUs to validate the correct connectivity and BFD Continuity Check (CC) PDUs to validate that the connection is available.

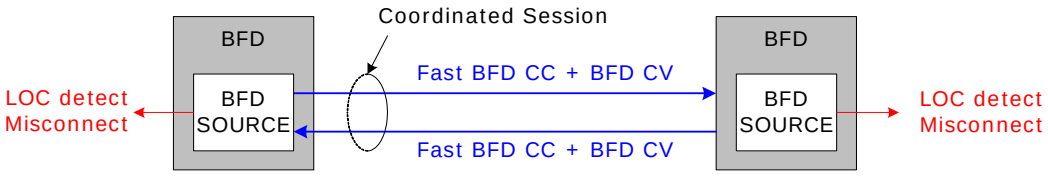
MPLS-TP BFD (RFC-6428) can run in Coordinated or independent mode.

In Coordinated mode, the BFD session consists of a single BFD entity which sends Fast BFD CC PDUs interleaved with Slow BFD CV PDUs towards the remote BFD MEP. The same frame flow is sent in the opposite direction by the remote MEP.

in this document, the BFD instance, which is used in Coordinated mode, is referred to as BFD source (BFD_SRC).

Coordinated mode is shown in the following illustration.

Figure 86 • **BFD Coordinated Mode**



The BFD source checks the incoming BFD CV PDU for mis-connectivity in software. If more than a configurable number of BFD CC PDUs are not received, the VOE signals loss of continuity (LOC).

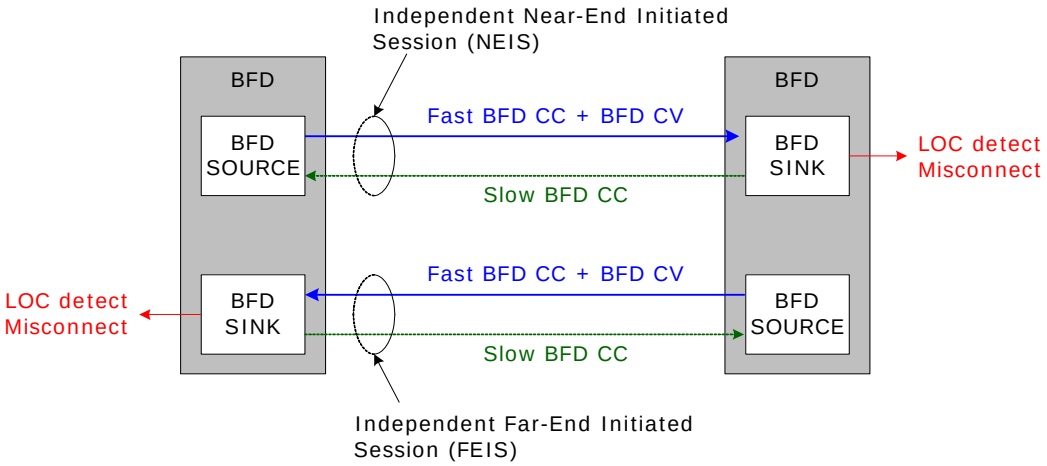
In Coordinated mode, there is only a single session between the BFD Peer MEPs. Within the scope of this document a BFD connection configured for Coordinated Mode is referred to as a Coordinated BFD Session.

In Independent mode, the BFD session consists of two BFD entities. Within the scope of this document, these two entities are referred to as follows:

- BFD Source (BFD_SRC): The BFD_SRC sends Fast BFD CC PDUs interleaved with Slow BFD CV PDUs towards the remote BFD_SINK. Slow BFD CC PDUs are received from the remote BFD_SINK.
- BFD Sink (BFD_SINK): The BFD_SINK sends Slow BFD CC PDUs towards the remote BFD_SRC. Fast BFD CC and Slow BFD CV are received from the remote BFD_SRC.

Independent mode is shown in the following illustration.

Figure 87 • **BFD Independent Mode**



In Independent mode, there are two sessions between the BFD Peer MEPs:

- **Local BFD_SRC and Remote BFD_SINK.** Within the scope of this document, this session is referred to as the Near-End Initiated Session (NEIS).
- **Local BFD_SINK and Remote BFD_SRC.** Within the scope of this document, this session is referred to as the Far-End Initiated Session (FEIS).

In Independent mode, only the BFD_SINK checks CV PDUs for misconnectivity and loss of continuity (LOC). Within the scope of this document, a BFD connection configured for Independent mode is referred to as an independent BFD session.

A single VOE can support either Coordinated BFD mode or Independent BFD mode, depending on configuration.

3.25.3.4 VOE Implementation

Each VOE implements separate functions for BFD_SRC and BFD_SINK. The registers belonging to each of the sessions are named with the following subscript:

- *_SRC - Register belongs to the BFD_SRC function
- *_SINK - Register belongs to the BFD_SINK function

When the VOE is configured for Coordinated BFD mode only, the BFD_SRC function is used. When configured for Independent BFD mode, both the BFD_SRC and the BFD_SINK functions are used.

For both the BFD_SRC and the BFD_SINK function, the VOE validates the received BFD PDUs. Valid BFD CC/CV PDUs clear the BFD LOC counter. BFD CV PDUs must be extracted to the CPU for Connection Verification in a CPU.

The VOE implements an LOC counter, which is updated as follows:

- Coordinated BFD mode: The LOC counter is incremented with a rate determined by the expected BFD CC Rx interval in the BFD Source. The LOC counter is incremented by the LOCC. For more information, see Loss of Continuity Controller (LOCC), page 221.
  The LOC counter is cleared every time a valid BFD CC or CV PDU is received by the BFD source. If the LOC counter reaches the detect multiplier value configured for the BFD source, the VOE generates an LOC event.
  In Coordinated mode, the BFD source generates the LOC event.

- Independent BFD mode: The LOC counter is incremented with a rate determined by the expected BFD CC Rx interval in the BFD_SINK. The LOC counter is incremented by the LOCC.
  The LOC counter is cleared every time a valid BFD CC or CV PDU is received by the BFD_SINK.
  If the LOC counter reaches the detect multiplier value configured for the BFD_SINK, the VOE generates an LOC event.
  In Independent mode, only the BFD_SINK can generate an LOC event.

3.25.4 BFD Configuration

This section provides information about the processing and register configurations for Bidirectional Fowarding Detection (BFD).

3.25.4.1 Basic BFD Configuration

To enable processing of the BFD PDUs, the BFD-CC and BFD-CV PDUs must be enabled in BFD-CC (VOP_MPLS:VOE_CONF_MPLS:OAM_HW_CTRL_MPLS.BFD_CC_ENA) and BFD-CV (VOP_MPLS:VOE_CONF_MPLS:OAM_HW_CTRL_MPLS.BFD_CV_ENA).

The G-ACH Channel Type used in the BFD CC PDUs is configurable in VOP_MPLS:VOE_CONF_MPLS:BFD_CONFIG.BFD_CC_RFC6428.

3.25.4.2 BFD-CC/BFD-CV Counter Configuration

The OAM MPLS-TP PDU Tx/Rx counters are updated according to the counter configuration:

- BFD-CC: VOP_MPLS:VOE_CONF_MPLS:OAM_CNT_SEL_MPLS.BFD_CC_CNT_SEL_ENA
- BFD-CV: VOP_MPLS:VOE_CONF_MPLS:OAM_CNT_SEL_MPLS.BFD_CV_CNT_SEL_ENA

3.25.4.3 Coordinated BFD Mode

In Coordinated BFD mode, only the BFD_SRC is configured. Note that the BFD_SRC registers are also used in Independent BFD Mode; however, the value configured into the registers change slightly when the VOE is programmed for Independent BFD mode.

Within this section, the term remote BFD entity refers to the remote BFD_SRC.

The VOE is configured for Coordinated BFD mode by setting VOP_MPLS:VOE_CONF_MPLS:BFD_CONFIG.BFD_COORDINATED_MODE_ENA to 1.

The following registers are programmed by software and never altered by the VOE:

- BFD discriminator of the local BFD_SRC (VOP_MPLS:VOE_CONF_MPLS:BFD_LOCAL_DISCR_SRC.BFD_LOCAL_DISCR_SRC).
- BFD discriminator of the remote BFD entity (VOP_MPLS:VOE_CONF_MPLS:BFD_REMOTE_DISCR_SRC.BFD_REMOTE_DISCR_SRC).
- BFD state of the local BFD_SRC (VOP_MPLS:VOE_STAT_MPLS:BFD_SRC_INFO.BFD_LOCAL_STATE_SRC). This register is optionally written into all valid BFD Tx frames.
- BFD diagnostic code of the local BFD_SRC (VOP_MPLS:VOE_STAT_MPLS:BFD_SRC_INFO.BFD_LOCAL_DIAG_SRC). This register is optionally written into all valid BFD Tx frames.

BFD detect multiplier of the local BFD_SRC (VOP_MPLS:VOE_STAT_MPLS:BFD_SRC_INFO.BFD_LOCAL_DM_SRC). This register is optionally written into all valid BFD Tx frames.

The following registers are updated by the VOE if Rx sampling is enabled through VOP_MPLS:VOE_CONF_MPLS:BFD_CONFIG.BFD_RX_SAMPLE_ENA:

- BFD state of the remote BFD entity (VOP_MPLS:VOE_STAT_MPLS:BFD_SRC_INFO.BFD_REMOTE_STATE_SRC). This register is updated with the value of the STA field of the latest valid Rx BFD CC PDU.
- BFD diagnostic code of the remote BFD entity (VOP_MPLS:VOE_STAT_MPLS:BFD_SRC_INFO.BFD_REMOTE_DIAG_SRC). This register is updated with the value of the DIAG field of the latest valid Rx BFD CC PDU.

- BFD Detect Multiplier of the remote BFD entity (VOP_MPLS:VOE_STAT_MPLS:BFD_SRC_INFO.BFD_REMOTE_DM_SRC). This register is updated with the value of the DETECT MULT field of the latest valid Rx BFD CC PDU. In Coordinated BFD Mode this value is used for detecting LOC. When the LOC counter reaches this value, the VOE generates an LOC event.

3.25.4.4 Independent BFD Mode

When the VOE is configured for Independent BFD mode, the BFD_SRC registers must be configured as described in the previous section, with the exception that the term remote BFD entity now denotes the remote MEP BFD_SINK.

The VOE is configured for Independent BFD Mode by setting VOP_MPLS:VOE_CONF_MPLS:BFD_CONFIG.BFD_COORDINATED_MODE_ENA to 0.

Additionally the following registers are programmed by software and never altered by the VOE:

- BFD discriminator of the local BFD_SINK (VOP_MPLS:VOE_CONF_MPLS:BFD_LOCAL_DISCR_SINK.BFD_LOCAL_DISCR_SINK).
- BFD discriminator of the remote BFD_SINK (VOP_MPLS:VOE_CONF_MPLS:BFD_REMOTE_DISCR_SINK.BFD_REMOTE_DISCR_SINK).
- BFD state of the local BFD_SINK (VOP_MPLS:VOE_STAT_MPLS:BFD_SINK_INFO.BFD_LOCAL_STATE_SINK).This register is optionally written into all valid BFD Tx frames.
- BFD diagnostic code of the local BFD_SINK (VOP_MPLS:VOE_STAT_MPLS:BFD_SINK_INFO.BFD_LOCAL_DIAG_SINK). This register is optionally written into all valid BFD Tx frames.

BFD detect multiplier of the local BFD_SINK (VOP_MPLS:VOE_STAT_MPLS:BFD_SINK_INFO.BFD_LOCAL_DM_SINK). This register is optionally written into all valid BFD Tx frames.

The following registers are programmed by SW and are updated by the VOE if Rx sampling is enabled through VOP_MPLS:VOE_CONF_MPLS:BFD_CONFIG.BFD_RX_SAMPLE_ENA:

- BFD State of the remote BFD_SINK (VOP_MPLS:VOE_STAT_MPLS:BFD_SINK_INFO.BFD_REMOTE_STATE_SINK. This register is updated with the value of the STA field of the latest valid Rx BFD CC PDU.
- BFD Diagnostic code of the remote BFD_SINK (VOP_MPLS:VOE_STAT_MPLS:BFD_SINK_INFO.BFD_REMOTE_DIAG_SINK). This register is updated with the value of the DIAG field of the latest valid Rx BFD CC PDU.
- BFD Detect Multiplier of the remote BFD_SINK (VOP_MPLS:VOE_STAT_MPLS:BFD_SINK_INFO.BFD_REMOTE_DM_SINK. This register is updated with the value of the DETECT MULT field of the latest valid Rx BFD CC PDU. In Independent BFD mode, this value is used for detecting LOC. When the LOC counter reaches this value, an LOC event is generated.

3.25.5 BFD Frame Reception

When a BFD Rx PDU is received by the VOE, it is optionally verified by the VOE. The BFD Rx frame validation is based on RFC-5880, section 6.8.6. In addition, a single check is added by RFC-6428. For more information about Rx validation, see BFD Rx Validation, page 278.

The frame validation determines if the received BFD PDU is a valid PDU or not. If it is valid. If the Rx PDU is not valid, the frame is discarded. A sticky bit is set to identify the discard reason. For more information about the extraction of valid and invalid BFD PDUs, see BFD Rx Frame Extraction, page 281.

3.25.5.1 BFD Rx Validation

In this section, Rx_frm.<name> refers to the field <name> in the received BFD frame.

Rx frames are subject to a number of Rx checks described in the sections. The checks are applied in the order in which they are described. If an Rx BFD PDU fails one of the checks, further processing is suspended. Hence a frame can only fail one of the Rx checks.

- Version check. The supported BFD version is programmed into the VOP in VOP::VERSION_CTRL_MPLS.BFD_VERSION (3 bit). Upon reception, the Rx_frm.version is validated against the above register. If the values do not match, the BFD frame is invalid and the VOP_MPLS:VOE_STAT_MPLS:BFD_RX_STICKY.VERSION_ERR_STICKY sticky bit is set. BFD PDUs failing the version check can optionally be extracted to CPU using VOP_MPLS:VOE_STAT_MPLS:PDU_EXTRACT_MPLS.BFD_RX_ERR_EXTR.
- Minimum length (MIN_LEN) check. If Rx_frm.A (Authentication Present) is cleared, then the Rx_frm.length field must be greater or equal to 24 bytes.
  - If Rx_frm.A (Authentication Present) is set, then the Rx_frm.length field must be greater or equal to 26 bytes.
  - If the Rx_frm.length does not match the above expectations, the frame is marked as invalid and the VOP_MPLS:VOE_STAT_MPLS:BFD_RX_STICKY.MIN_LEN_ERR_STICKY sticky bit is set. BFD PDUs failing the minimum length check can optionally be extracted to CPU using VOP_MPLS:VOE_STAT_MPLS:PDU_EXTRACT_MPLS.BFD_RX_ERR_EXTR.
- Maximum length (MAX_LEN) check. The BFD maximum length is configured in VOP_MPLS:VOE_CONF_MPLS.BFD_CONFIG.BFD_MAX_LEN. The Rx_frm.length of the incoming BFD PDUs is verified against this value. If the Rx_frm.length is larger than the configured value the frame is marked as invalid and the VOP_MPLS:VOE_STAT_MPLS:BFD_RX_STICKY.MAX_LEN_ERR_STICKY sticky is set. BFD PDUs failing the maximum length check can optionally be extracted to CPU using VOP_MPLS:VOE_STAT_MPLS:PDU_EXTRACT_MPLS.BFD_RX_ERR_EXTR.
- Detect zero multiplier (DM_ZERO) check. If Rx_frm.DetectMult is zero, the frame is marked as invalid, and theVOP_MPLS:VOE_STAT_MPLS:BFD_RX_STICKY.DM_ZERO_ERR_STICKY sticky bit is set. BFD PDUs failing the detect zero multiplier check can optionally be extracted to CPU using VOP_MPLS:VOE_STAT_MPLS:PDU_EXTRACT_MPLS.BFD_RX_ERR_EXTR.
- Multipoint set (M_BIT_SET) check. If the Rx_frm.M (multipoint) is set, the frame is marked as invalid and the VOP_MPLS:VOE_STAT_MPLS:BFD_RX_STICKY.M_BIT_SET_ERR_STICKY sticky bit is set. BFD PDUs failing the multipoint set check can optionally be extracted to CPU using VOP_MPLS:VOE_STAT_MPLS:PDU_EXTRACT_MPLS.BFD_RX_ERR_EXTR.
- Your discriminator (YOUR_DISCR) check. The VOE is either configured for Coordinated or Independent BFD mode. If configured for Coordinated BFD mode, the Rx_frm.YourDiscriminator is validated against the discriminator of the local BFD_SRC using VOP_MPLS:VOE_CONF_MPLS:BFD_LOCAL_DISCR_SRC.BFD_LOCAL_DISCR_SRC. If there is a match, the PDU is considered part of the Coordinated BFD session.

  If configured for Independent BFD mode, the Rx_frm.YourDiscriminator is validated against the following discriminators:
  NEIS (VOP_MPLS:VOE_CONF_MPLS:BFD_LOCAL_DISCR_SRC.BFD_LOCAL_DISCR_SRC).
  FEIS (VOP_MPLS:VOE_CONF_MPLS:BFD_LOCAL_DISCR_SINK.BFD_LOCAL_DISCR_SINK).

  If there is a match, the PDU is considered part of the corresponding NEIS/FEIS session.
  If there is no match, the frame is marked as invalid, and the VOP_MPLS:VOE_STAT_MPLS:BFD_RX_STICKY.YOUR_DISCR_ERR_STICKY sticky bit is set. BFD PDUs failing the YourDiscriminator check can optionally be extracted to CPU using VOP_MPLS:VOE_STAT_MPLS:PDU_EXTRACT_MPLS.BFD_RX_ERR_EXTR.

  If the YOUR_DISCR check is disabled, the frame is assumed to belong to the Coordinated/NEIS session.
- My discriminator (MY_DISCR) check: The VOE implementation assumes that the CPU has correctly detected the remote end discriminator value (or values) and that it has informed the remote end of the local discriminator value (or values). As a result, the my discriminator check requires the Rx_frm.MyDiscriminator to match one of the configured values.
  The VOE is configured for either Coordinated- or Independent BFD session. If configured for Coordinated BFD Mode, the Rx_frm.MyDiscriminator is validated against the Discriminator of the remote BFD_SRC function in VOP_MPLS:VOE_CONF_MPLS:BFD_REMOTE_DISCR_SRC.BFD_REMOTE_DISCR_SRC.
  If configured for Independent BFD mode, the Rx_frm.MyDiscriminator is validated depending on whether the your discriminator check determined the PDU as belonging to the NEIS or the FEIS session.

NEIS:
VOP_MPLS:VOE_CONF_MPLS:BFD_REMOTE_DISCR_SRC.BFD_REMOTE_DISCR_SRC
FEIS:
VOP_MPLS:VOE_CONF_MPLS:BFD_REMOTE_DISCR_SINK.BFD_REMOTE_DISCR_SINK.

If the Rx_frm.MyDiscriminator fails the my discriminator check, the frame is marked as invalid and the sticky bit VOP_MPLS:VOE_STAT_MPLS:BFD_RX_STICKY.MY_DISCR_ERR_STICKY is set. BFD PDUs failing the my discriminator check can optionally be extracted to CPU using VOP_MPLS:VOE_STAT_MPLS:PDU_EXTRACT_MPLS.BFD_RX_ERR_EXTR.

- Authentication mismatch (AUTH_MISMATCH) check. If authentication is enabled, the VOE supports validating the Rx_frm.A authentication bit against the value configured for the BFD session. The VOE enables authentication independently for the supported PDU types:
  BFD-CC: VOP_MPLS:VOE_CONF_MPLS:BFD_CONFIG.BFD_CC_AUTH_ENA
  BFD-CV: VOP_MPLS:VOE_CONF_MPLS:BFD_CONFIG.BFD_CV_AUTH_ENA

  Upon reception, the Rx_frm.A is validated against the value configured for the PDU type (CC/CV). If the Rx_frm.A does not match the configured value, the frame is marked as invalid, and the VOP_MPLS:VOE_STAT_MPLS:BFD_RX_STICKY.AUTH_MISMATCH_ERR_STICKY sticky bit is set. BFD PDUs failing the authentication mismatch check can optionally be extracted to CPU using VOP_MPLS:VOE_STAT_MPLS:PDU_EXTRACT_MPLS.BFD_RX_ERR_EXTR.

  Note that BFD PDUs, for which authentication is enabled, all subsequent processing must be done by the CPU, because the VOE does validate the authentication itself.
- Demand bit set (D_BIT_SET) check. If the Rx_frm.D demand bit is set, the frame is marked as invalid, and the VOP_MPLS:VOE_STAT_MPLS:BFD_RX_STICKY.D_BIT_SET_ERR_STICKY sticky bit is set. BFD PDUs failing the demand bit set check can optionally be extracted to CPU using VOP_MPLS:VOE_STAT_MPLS:PDU_EXTRACT_MPLS.BFD_RX_ERR_EXTR.
- Poll and final bits set (P_AND_F_BIT_SET) check. If both the Rx_frm.P poll bit and the Rx_frm.F final bit are set, the frame is marked as invalid, and the VOP_MPLS:VOE_STAT_MPLS:BFD_RX_STICKY.P_AND_F_BIT_SET_ERR_STICKY sticky bit is set. BFD PDUs failing the poll and final bits set check can optionally be extracted to CPU using VOP_MPLS:VOE_STAT_MPLS:PDU_EXTRACT_MPLS.BFD_RX_ERR_EXTR.

Portions of the Rx validation can be disabled using the following configuration:

- Enable BFD CC validation (VOP_MPLS:VOE_CONF_MPLS:BFD_CONFIG.BFD_RX_VERIFY_CC_ENA). If not enabled, all checks are disabled for BFD CC PDUs.
- Enable BFD CV validation (VOP_MPLS:VOE_CONF_MPLS:BFD_CONFIG.BFD_RX_VERIFY_CV_ENA).If not enabled, all checks are disabled for BFD CV PDUs.
- Enable version check (VOP_MPLS:VOE_CONF_MPLS:BFD_CONFIG.BFD_RX_VERIFY_VERSION_ENA). If not enabled, the VERSION check is disabled for BFD (CC/CV) PDUs.
- Enable minimum length check (VOP_MPLS:VOE_CONF_MPLS:BFD_CONFIG.BFD_RX_VERIFY_MIN_LEN_ENA). If not enabled, the MIN_LEN check is disabled for BFD (CC/CV) PDUs.
- Enable discriminator checks (VOP_MPLS:VOE_CONF_MPLS:BFD_CONFIG.BFD_RX_VERIFY_DISCR_ENA). If not enabled, the my discriminator and your discriminator checks are disabled for BFD (CC/CV) PDUs. If these checks are disabled, the frame is assumed to belong to the Coordinated/NEIS session.
- Enable BFD flag checks (VOP_MPLS:VOE_CONF_MPLS:BFD_CONFIG.BFD_RX_VERIFY_FLAGS_ENA). If not enabled, the detect zero multiplier check, the multiplier set check, the authentication mismatch check, the demand bit set check, and the poll and final bits set check are disabled for BFD CC/CV PDUs.

3.25.5.2 BFD Valid Rx Frames

Upon reception of a valid BFD-CC PDU, the VOE performs the following actions:

- Clears the LOC counter (VOP_MPLS:VOE_STAT_MPLS:BFD_STAT.BFD_MISS_CNT).

- Updates the VOP_MPLS:VOE_STAT_MPLS:CPT_RX_STICKY_MPLS.BFD_CC_RX_STICKY bit.

Depending on the configuration of the following bit field, the VOE samples the BFC CC PDU and updates a number of registers:

If BFC-CC sampling is enabled (VOP_MPLS:VOE_CONF_MPLS:BFD_CONFIG.BFD_RX_SAMPLE_ENA), the VOE updates the following fields:

- Samples the Rx_frm.State and updates:
  Coordinated Session or NEIS
  (VOP_MPLS:VOE_STAT_MPLS:BFD_SRC_INFO.BFD_REMOTE_STATE_SRC)
  FEIS (VOP_MPLS:VOE_STAT_MPLS:BFD_SINK_INFO.BFD_REMOTE_STATE_SINK)
- Samples the Rx_frm.Diag and updates:
  Coordinated Session or NEIS
  (VOP_MPLS:VOE_STAT_MPLS:BFD_SRC_INFO.BFD_REMOTE_DIAG_SRC)
  FEIS (VOP_MPLS:VOE_STAT_MPLS:BFD_SINK_INFO.BFD_REMOTE_DIAG_SINK)
- Samples the Rx_frm.DetectMult and updates:
  Coordinated Session or NEIS
  (VOP_MPLS:VOE_STAT_MPLS:BFD_SRC_INFO.BFD_REMOTE_DM_SRC)
  FEIS (VOP_MPLS:VOE_STAT_MPLS:BFD_SINK_INFO.BFD_REMOTE_DM_SINK).

The VOE extracts selected fields from the Rx BFD PDU and compares the new values to the values configured in the VOE. If a change is detected, the following sticky bits are asserted.

- Coordinated or NEIS session:
  VOP_MPLS:VOE_STAT_MPLS:INTR_STICKY_MPLS.BFD_RX_DM_CHANGE_SRC_STICKY
  VOP_MPLS:VOE_STAT_MPLS:INTR_STICKY_MPLS.BFD_RX_STATE_CHANGE_SRC_STICKY
  VOP_MPLS:VOE_STAT_MPLS:INTR_STICKY_MPLS.BFD_RX_DIAG_CHANGE_SRC_STICKY

  If the P-flag is asserted in the Rx BFD PDU, a sticky bit is asserted (VOP_MPLS:VOE_STAT_MPLS:INTR_STICKY_MPLS.BFD_RX_P_SET_SRC_STICKY).

  If the F-flag is asserted in the Rx BFD PDU, a sticky bit is asserted (VOP_MPLS:VOE_STAT_MPLS:INTR_STICKY_MPLS.BFD_RX_F_SET_SRC_STICKY).
- FEIS session:
  VOP_MPLS:VOE_STAT_MPLS:INTR_STICKY_MPLS.BFD_RX_DM_CHANGE_SINK_STICKY
  VOP_MPLS:VOE_STAT_MPLS:INTR_STICKY_MPLS.BFD_RX_STATE_CHANGE_SINK_STICKY
  VOP_MPLS:VOE_STAT_MPLS:INTR_STICKY_MPLS.BFD_RX_DIAG_CHANGE_SINK_STICKY

  If the P-flag is asserted in the Rx BFD PDU, a sticky bit is asserted (VOP_MPLS:VOE_STAT_MPLS:INTR_STICKY_MPLS.BFD_RX_P_SET_SINK_STICKY).

  If the F-flag is asserted in the Rx BFD PDU, a sticky bit is asserted (VOP_MPLS:VOE_STAT_MPLS:INTR_STICKY_MPLS.BFD_RX_F_SET_SINK_STICKY).

The above sticky bits can generate an interrupt if the source in VOP_MPLS:VOE_STAT_MPLS:INTR_ENA_MPLS.

Upon receiving a valid BFD-CV PDU, the VOE performs the following:

- Clears the LOC counter (VOP_MPLS:VOE_STAT_MPLS:BFD_STAT.BFD_MISS_CNT).
- Updates the following sticky bits:
  VOP_MPLS:VOE_STAT_MPLS:CPT_RX_STICKY_MPLS.BFD_CV_RX_STICKY

  If the P-flag is asserted in the Rx BFD PDU, a sticky bit is asserted,
  VOP_MPLS:VOE_STAT_MPLS:INTR_STICKY_MPLS.BFD_RX_P_SET_SINK_STICKY

  If the F-flag is asserted in the Rx BFD PDU, a sticky bit is asserted:
  VOP_MPLS:VOE_STAT_MPLS:INTR_STICKY_MPLS.BFD_RX_F_SET_SINK_STICKY

3.25.5.3 BFD Rx Frame Extraction

The VOE can be configured to extract BFD PDUs based on the following reasons:

- All valid Rx BFD CC PDUs (VOP_MPLS:VOE_CONF_MPLS:CPU_COPY_CTRL_MPLS.BFD_CC_CPU_COPY_ENA).
- All valid Rx BFD CV PDUs (VOP_MPLS:VOE_CONF_MPLS:CPU_COPY_CTRL_MPLS.BFD_CV_CPU_COPY_ENA).
- Next valid Rx BFD CC PDUs (VOP_MPLS:VOE_STAT_MPLS:PDU_EXTRACT_MPLS.BFD_CC_RX_NEXT_GOOD_EXTR).
- Next valid Rx BFD CV PDUs (VOP_MPLS:VOE_STAT_MPLS:PDU_EXTRACT_MPLS.BFD_CV_RX_NEXT_GOOD_EXTR).

The following frame extraction extracts either the next PDU (Hit Me Once) or all the PDUs matching the extract criterion, depending on the configuration in VOP_MPLS:VOE_STAT_MPLS:PDU_EXTRACT_MPLS.EXTRACT_HIT_ME_ONCE.

Frames can be extracted based on the following criteria:

- Valid Rx BFD CC with an Rx parameter change (VOP_MPLS:VOE_STAT_MPLS:PDU_EXTRACT_MPLS.BFD_RX_PARAM_CHANGE_EXTR) Extract Rx frames in which one of the following BFD parameters changed compared to the previous BFD PDU belonging to the same session (Coordinated/NEIS/FEIS):
  - BFD.DM
  - BFD.DIAG
  - BFD.STATE
- Valid Rx BFD (CC/CV) with the Poll bit set (VOP_MPLS:VOE_STAT_MPLS:PDU_EXTRACT_MPLS.BFD_RX_P_SET_EXTR).
- Valid Rx BFD (CC/CV) with the final bit set (VOP_MPLS:VOE_STAT_MPLS:PDU_EXTRACT_MPLS.BFD_RX_F_SET_EXTR).
- Invalid Rx BFD (CC/CV) discarded by Rx validation (VOP_MPLS:VOE_STAT_MPLS:PDU_EXTRACT_MPLS.BFD_RX_ERR_EXTR).

All valid PDUs are extracted to the BFD CC/CV queue depending on the PDU type. All invalid PDUs are extracted to the CPU error queue.

3.25.6 BFD Frame Transmission

The VOE does not control the frame transmission rate of BFD frames. This must be handled by a CPU or the AFI. The VOE only updates Tx statistics and optionally modifies the Tx PDU as part of a VOE Tx lookup.

3.25.6.1 BFD Tx Validation

In this section Tx_frm.<name> refers to the field <name> in the outgoing BFD frame.

If configured to update the Tx BFD PDU, the VOE determines which session the Tx PDU belongs to. To do this, the VOE matches the Tx_frm.MyDiscriminator against the discriminators configured in the VOE. If configured for Coordinated BFD Mode the VOE matches the Tx_frm.MyDiscriminator against VOP_MPLS:VOE_CONF_MPLS:BFD_LOCAL_DISCR_SRC.BFD_LOCAL_DISCR_SRC. In case of a match the Tx PDU is part of the Coordinated Session. If configured for Independent BFD Mode the VOE matches the Tx_frm.MyDiscriminator against the following registers:

- NEIS: VOP_MPLS:VOE_CONF_MPLS:BFD_LOCAL_DISCR_SRC.BFD_LOCAL_DISCR_SRC FEIS: VOP_MPLS:VOE_CONF_MPLS:BFD_LOCAL_DISCR_SINK.BFD_LOCAL_DISCR_SINK

If there is a match, the Tx PDU is part of either a NEIS or a FEIS session.

If the Tx_frm.MyDiscriminator does not match any of the above values, the PDU is discarded and the sticky bit is VOP_MPLS:VOE_STAT_MPLS:BFD_TX_STICKY.TX_MY_DISCR_MISMATCH is set. Tx BFD PDUs failing the above check can optionally be extracted to CPU using VOP_MPLS:VOE_STAT_MPLS:PDU_EXTRACT_MPLS.BFD_TX_ERR_EXTR. PDUs are be extracted to the CPU error queue.

Frames are extracted either Hit-Me-Once or all failing PDUs (VOP_MPLS:VOE_STAT_MPLS:PDU_EXTRACT_MPLS.EXTRACT_HIT_ME_ONCE).

3.25.6.2 Updating Tx BFD PDUs

The Tx PDU updating is enabled by VOP_MPLS:VOE_CONF_MPLS:BFD_CONFIG.BFD_TX_UPDATE_ENA. If Tx updating is enabled the following fields are updated for all Tx BFD PDUs (CC/CV). The value written to the field depends on which session the Tx PDU belongs to:

- BFD.Diag:
  Coordinated Session or NEIS:
  VOP_MPLS:VOE_STAT_MPLS:BFD_SRC_INFO.BFD_LOCAL_DIAG_SRC
  FEIS:
  VOP_MPLS:VOE_STAT_MPLS:BFD_SINK_INFO.BFD_LOCAL_DIAG_SINK
- BFD.State:
  Coordinated Session or NEIS:
  VOP_MPLS:VOE_STAT_MPLS:BFD_SRC_INFO.BFD_LOCAL_STATE_SRC
  FEIS:
  VOP_MPLS:VOE_STAT_MPLS:BFD_SINK_INFO.BFD_LOCAL_STATE_SINK
- BFD.DM:
  Coordinated Session or NEIS:
  VOP_MPLS:VOE_STAT_MPLS:BFD_SRC_INFO.BFD_LOCAL_DM_SRC
  FEIS:
  VOP_MPLS:VOE_STAT_MPLS:BFD_SINK_INFO.BFD_LOCAL_DM_SINK

3.25.7 BFD VOE Functions

This section describes miscellaneous BFD functionality.

3.25.7.1 Loss of Continuity (LOC) Detection

The VOE implements support for LOC detection for BFD sessions. When an LOC condition is detected, the VOE asserts a sticky bit and optionally generates an interrupt.

The LOC mechanism is based on the BFD miss counter, counting the number of missing consecutive BFD PDUs from the remote BFD MEP and signaling LOC when the counter reaches a configurable value.

The BFD miss counter is incremented by the LOCC every time a BFD frame is expected and cleared every time a valid BFD PDU is received.

The current BFD miss counter value is available in VOP_MPLS:VOE_STAT_MPLS:BFD_STAT.BFD_MISS_CNT.

The BFD miss counter is updated every time the selected LOC timer expires. The selected LOC timer to update the BFD miss counter is configured in VOP_MPLS:VOE_CONF_MPLS:BFD_CONFIG.BFD_SCAN_PERIOD.

Due to the way the LOC scan is implemented for Y.1731, the BFD miss counter is updated with twice the frequency that the BFD Rx interval dictates. For more information, see Loss of Continuity Controller, page 233. This implies that if the LOC detect signal should detect 3 missing BFD frames, the LOC detect must be signaled when the BFD miss counter equals 6. For this reason the BFD miss counter is one bit wider than the size of the Detect Multiplier.

A BFD session signals LOC detect when the number of consecutive missing BFD frames is equal to the session detect multiplier. For BFD MPLS-TP the detect multiplier must be fixed to 3. The VOE signals LOC detect when the BFD miss counter reaches the sessions detect multiplier. The detect multiplier used to signal LOC detect depends on the configuration of the BFD session:

- Coordinated BFD mode
  (VOP_MPLS:VOE_STAT_MPLS:BFD_SRC_INFO.BFD_REMOTE_DM_SRC).
- Independent BFD mode
  (VOP_MPLS:VOE_STAT_MPLS:BFD_SINK_INFO.BFD_REMOTE_DM_SINK).

The VOE stores the current LOC state in VOP_MPLS:VOE_STAT_MPLS.BFD_RX_LAST.BFD_LOC_DEFECT. When LOC status changes, the VOE signals the LOC change event by asserting the

VOP_MPLS:VOE_STAT_MPLS:INTR_STICKY_MPLS.BFD_LOC_CHANGE_STICKY bit. The VOE optionally generates an interrupt when the LOC event is detected using VOP_MPLS:VOE_STAT_MPLS:INTR_ENA_MPLS.BFD_LOC_CHANGE_INT_ENA.

3.25.7.2 BFD Interrupt Sources

A number of events can optionally generate an interrupt in the VOE. Each of the events which can generate an interrupt is represented by a sticky bit in the register VOP_MPLS:VOE_STAT_MPLS:INTR_STICKY_MPLS. When one of these sticky bits is asserted, it generates an interrupt, if the corresponding interrupt is enabled in the register VOP_MPLS:VOE_STAT_MPLS:INTR_ENA_MPLS.

3.25.7.3 Misconnectivity Event

The VOE validates Rx BFD CV PDUs and extracts only valid BFD CV PDUs to the CPU. The CPU must validate that the Rx BFD CV PDU is received from the expected Peer MEP. If this is not the case, the CPU must update the BFD state condition and take other required measures.

3.25.7.4 BFD Poll Sequence

The VOE can optionally extract valid Rx BFD PDUs with either the P-bit or the F-bit set to the CPU, where the CPU must handle the Poll/Final sequence when negotiating BFD parameters at session startup.

3.25.8 BFD Statistics

This section lists the MPLS-TP related counters provided by the VOE.

3.25.8.1 Rx Statistics

The following 32-bit Rx counters are supported:

- Number of CC valid Rx frames (VOP_MPLS:VOE_STAT_MPLS:BFD_CC_RX_VLD_CNT_REG.BFD_CC_RX_VLD_CNT).
- Number of CV valid Rx frames (VOP_MPLS:VOE_STAT_MPLS:BFD_CV_RX_VLD_CNT_REG.BFD_CV_RX_VLD_CNT).
- Number of CC invalid Rx frames (VOP_MPLS:VOE_STAT_MPLS:BFD_CC_RX_INVLD_CNT_REG.BFD_CC_RX_INVLD_CNT).
- Number of CV invalid Rx frames (VOP_MPLS:VOE_STAT_MPLS:BFD_CV_RX_INVLD_CNT_REG.BFD_CV_RX_INVLD_CNT).

3.25.8.2 Tx Statistics

The following 32-bit Tx counters are supported:

- CC Tx frames (VOP_MPLS:VOE_STAT_MPLS:BFD_CC_TX_CNT_REG.BFD_CC_TX_CNT).
- CV Tx frames (VOP_MPLS:VOE_STAT_MPLS:BFD_CV_TX_CNT_REG.BFD_CV_TX_CNT).

3.26 Shared Queue System and Hierarchical Scheduler

The device includes a shared queue system with a per-port ingress queue and 3,358 shared egress queues. The shared queue system has 8 megabits of buffer. It receives frames from 15 ports, stores them in a shared packet memory, and transmits them towards 15 destination ports. The first 11 ports are assigned directly to a specific front port, whereas the last four ports are internal ports.

Table 129 • **Port Definitions in Shared Queue System**

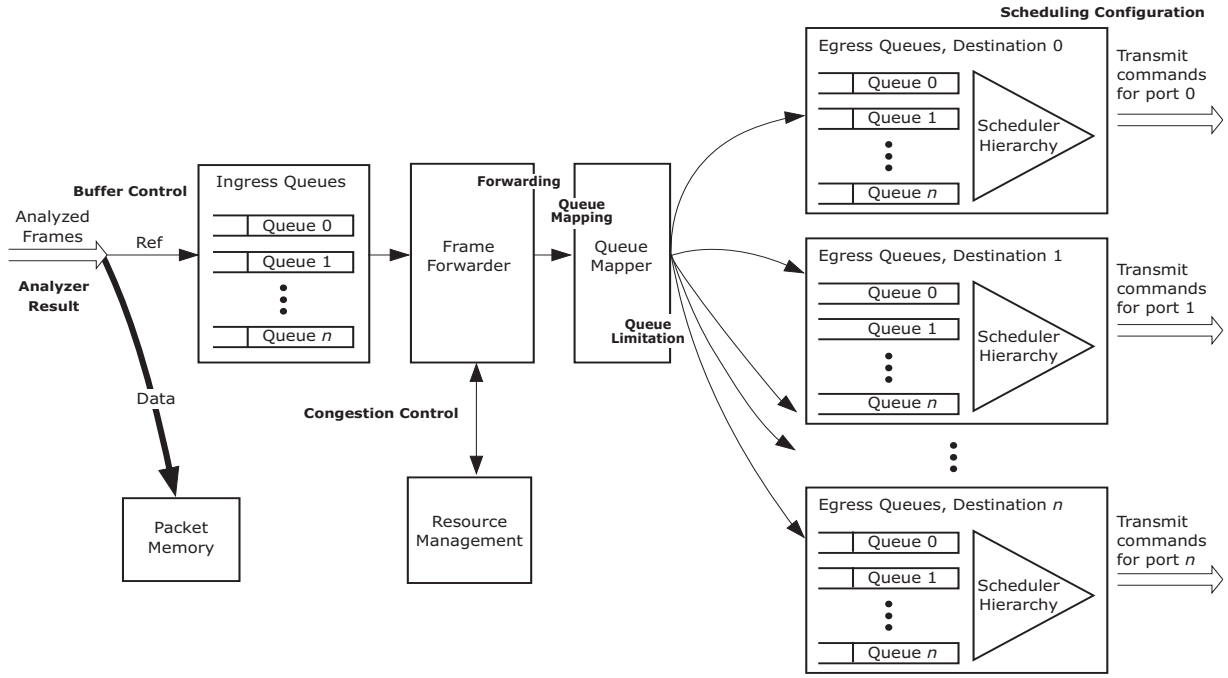
Ports	Connection
0 – 8	Front port, maximum bandwidth is 1 Gbps or 2.5 Gbps.
9 – 10	Front port, maximum bandwidth is 10 Gbps.
11 – 12	Frame injection and extraction CPU ports. The CPU connects these ports to DMA logic or to a register interface.

Table 129 • **Port Definitions in Shared Queue System *(continued)***

Ports	Connection
13	IP multicast loopback port (VD0). When the queue system is requested to transmit to VD0, it generates multiple copies. All copies are, outside the queue system, looped back into the analyzer. The rewriter changes the ingress port number to VD0 before the loop. The analyzer generates a routed version of each of the frame copies coming from VD0.
14	AFI loopback port (VD1). When the queue system is requested to send to VD1, it sends the request to the AFI. The AFI stores the request and can be configured to transmit the frame to any port in a programmed schedule. The AFI can transmit the frame to any of the egress ports. If VD1 is selected as destination by the AFI, the frame is looped back into the analyzer.

The following illustration provides an overview of shared queue system.

Figure 88 • **Shared Queue System Block Diagram**



Frames are stored in the packet memory and linked into the ingress queue for the logical source port along with the analyzer result.

The analyzer result is then processed by a frame forwarder, frame by frame. Each of the frame transmission requests are added to an egress queue in the queue system attached to a specific egress port. A transmission request can be a normal switching request to another port, a mirror copy, for stacking updates, and for a CPU connected to one of the egress ports. For VD0, it can do multiple copies of the request. The frame forwarder can also request that a specific frame copy be discarded.

The frame forwarder is efficient, because only frame references are switched, giving a forwarding bandwidth in the range of 67 Gbps (64 byte frames) to 2 terabits (Tbps) (1,518 byte frames). The frame data itself is not moved within the queue system.

The forwarding request is passed through a queue mapper, which then selects an egress queue based on the classified properties of the frame. For example, a simple switch may use the classified QoS class only and an advanced carrier switch may use MPLS-classified traffic class (TC).

A congestion control system can decide to discard some or all copies of the frame. This decision is based on consumption per QoS level. The queue mapper can be configured to change the drop decision using a congestion control mechanism based on queue sizes rather than only QoS class.

Each destination port has a part of the total shared queue pool attached. Transmission is scheduled towards the destination ports through a hierarchical scheduling system, where all queues belonging to each port are arbitrated. On the way out, frames pass through the rewriter where the frame data can be modified due to, for instance, routing, MPLS encapsulation, or tagging. The queue system does not change the original frame data.

An ingress port operates in one of three basic modes: drop mode, port flow control, or priority-based flow control. The following table lists the modes and flow definitions.

Table 130 • **Ingress Port Modes**

Mode	Flow
Drop mode (DROP)	The link partner is unaware of congestion and all frames received by the port are evaluated by the congestion control function where copies of the frame can be discarded.
Port flow control (FC)	The link partner is informed through pause frames (full-duplex) or collisions (half-duplex) that the port cannot accept more data. If the link partner obeys the pause requests, no incoming data is discarded by the port. If the pause frames are disobeyed by the link partner, the buffer control function in the queue system discards data before it reaches the ingress queues.
Priority-based flow control (PFC)	The link partner is informed about which QoS classes in the queue system are congested and the link partner should stop scheduling more frames belonging to these QoS classes. If the pause frames are disobeyed by the link partner, the buffer control function in the queue system discards data before it reaches the ingress queues.

3.26.1 Analyzer Result

The analyzer provides required information for switching frames in the shared queue system. The following table lists the information provided and a general description of the purpose.

Table 131 • **Analyzer Result**

Group	Field	Function
Destination selection	Destination set	Set of ports to receive a normal switched copy.
Destination selection	CPU queue mask	Set of CPU queues to receive a copy.
Destination selection	Mirror	Requests mirror copy.
Destination selection	Learn all	Requests learn copy to stack ports.
Congestion control	Drop mode	This particular frame can be dropped even if ingress port is setup for flow control.
Congestion control	DP	Drop precedence level.
Congestion control	Priorities	Ingress resource priority of the frame. Four classified classes are provided from the analyzer: QoS, PCP, COSID, and TC. All values between 0 and 7.
Queue mapping	SP	Super priority frame.
Queue mapping	QGRP	Queue group for service-based egress queuing.
Statistics	OAM type	Service counters can be configured to only count certain OAM frame types.

3.26.2 Buffer Control

Frames are stored in a shared packet memory bank, which contains 5,952 words of 176 bytes each. A memory manager controls which words the packet writer uses to store a frame.

With proper configuration, the packet memory bank always has a free word available for incoming frames. There are configurable thresholds for triggering pause frame requests on ports enabled for flow control and for discarding frames due to too large memory use when for instance pause frames are disobeyed by the link partner.

Frames discarded at this stage are discarded without considering the frame's destination ports. This kind of discarding is called tail dropping. It is an indication of head-of-line blocking and is not usually desired. For information about how it can be avoided by properly configuring the congestion control system, see Congestion Control, page 290.

The pause frame and discard mechanisms are activated when a threshold for the individual port's memory use is reached and when a threshold for the total use of memory is reached.

The following table lists the configuration registers involved in the buffer control.

Table 132 • **Buffer Control Registers Overview**

Register	Description	Replication
QSYS::ATOP	Maximum allowed memory use for a port before tail dropping is activated.	Per port
QSYS::ATOP_TOT_CFG	Total memory use before tail dropping can be activated.	None
QSYS::PAUSE_CFG	Thresholds for when to start and stop port flow control based on memory use by the port.	Per port
QSYS::PAUSE_TOT_CFG	Total memory use before flow control is activated.	None
QFWD::SWITCH_PORT_MODE	Enable drop mode for a port.	Per port

The following table lists available status information.

Table 133 • **Buffer Status Registers Overview**

Register	Description	Replication
QSYS::MMGT_PORT_USE	Current consumption for a specific port	None
QSYS::MMGT_PORT_VIEW	Selection of port to show usage for	None
QSYS::MMGT	Number of free words in packet memory	None

3.26.3 Forwarding

The frame forwarder processes frames at the head of the per-port ingress queues by adding references to the egress queues for each of the frames' destination ports. The references points from the egress queues to the stored frames. The forwarder can add from 78 million to 156 million references per second.

The ingress queues are selected by a weighted round-robin search, where the weights are configured in QFWD::SWITCH_PORT_MODE.FWD_URGENCY.

Each frame is processed as either a drop-mode frame or a flow control frame, which influences how the congestion control works. For a drop-mode frame, the congestion control can discard frame copies while for a flow control frame, it lets the frame stay at the head of the ingress queue blocking further processing of frames from the port until the congestion situation is over. A frame is processed as a drop-mode frame if one of the following conditions is met:

- The ingress port is configured to be an ingress drop port.
- The egress port is configured to be an egress drop port.
- The frame itself is analyzed to be a drop frame.

In terms of discarding frames, the forwarder can work in an ingress-focused statistics mode or an egress-focused statistics mode. The mode is configured per ingress port in QFWD::STAT_CNT_CFG.

In the ingress-focused mode, a frame can be discarded towards multiple destinations simultaneously. The discard statistics is only incremented if the frame is discarded towards all destinations.

In the egress-focused mode, the discard statistics is incremented for each copy of the frame being discarded at the expense of the forwarder only discarding one frame copy per cycle. As a result, the forwarder can become oversubscribed. For example, if two 10G ports are multicasting 64-byte frames at wire-speed to ten destinations, the total amount of frame copies per second is 20 × 10 frames/(8 × (64 + 20) ns) = 1,523 million frames per second. The forwarder can at maximum process 156 million frames per second. As a consequence, the ingress queues fill up and tail dropping is inevitable.

The highlights of the modes are listed in the following table.

Table 134 • **Statistics Modes**

Statistics Mode	Operation
Ingress-focused	A frame switched to N ports and discarded to M ports will update the discard statistics for the ingress port by one, only if N = 0 and M >0. The forwarder will use N cycles to process the frame.
Egress-focused	A frame switched to N ports and discarded to M ports will update the discard statistics for the M egress ports by one. The forwarder will use N+M cycles to process the frame.

3.26.3.1 Forward Pressure

The worst-case switching capacity in the frame forwarder is 78 million frames per second. This corresponds to 52 Gbps line rate with minimum-sized frames.

However, if a large share of these frames have multiple destinations, and are all close to the 64 byte minimum frame size, the processing speed is insufficient to process the destination copy one at a time. The consequence is that the ingress queues start to fill. The ingress queues contain both high and low priority traffic, unicast, and multicast frames. To avoid head-of-line blocking effects from the ingress queues filling up and eventually leading to tail dropping, a forward pressure system is in place. It can discard frame copies based on the size of the ingress queues. Forward pressure is configured in the QSYS::FWD_PRESSURE registers, where each port is allowed only a certain number of copies per frame if the number of pending transmit-requests in the port's ingress queue exceeds a threshold.

The following table lists the registers associated with configuring forward pressure.

Table 135 • **Forward Pressure Configuration Registers Overview**

Register	Description	Replication
QSYS::FWD_PRESSURE.FWD_PRESSURE	Configures forwarding pressure limits per port.	Per port
QSYS::MMGT_IQ_STAT	Returns current ingress queue size.	Per port
QFWD::FWD_PRESSURE.FWD_PRESS_DROP_CNT	Counts number of frame copies discarded due to this feature. Value is total discards for all ports in common.	None

3.26.3.2 Destination Selection

The forwarder adds references to the egress queues by evaluating the destination selection information shown in the following table from the analyzer.

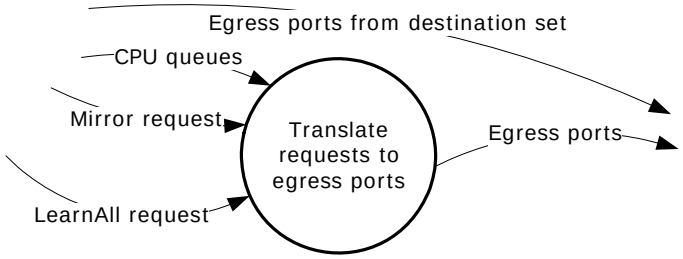
Table 136 • **Analyzer Destination Selection**

Field	Function
Destination set	Set of ports to receive a normal switched copy. All ports except the internal extraction ports have a bit in this mask.
CPUQ	Set of CPU queues to receive a copy.
Mirror	Request mirror copy.
Learn all	Request learn copy to stack ports.

The destination set selects specific egress ports, and the CPUQ, mirror, and learn-all fields are indirect frame destinations and must be translated to transmit requests through the frame copy configuration table. The table provides a specific egress port and QoS class to use in congestion control. The table is accessed through the QFWD::FRAME_COPY_CFG register.

The following illustration shows the translation of transmit requests.

Figure 89 • **Translation of Transmit Requests**



The CPUQ field instructs which of the eight CPU extraction queues should get a copy of this frame. Each queue in the queue system is translated into either one of the two internal CPU ports or to a front port. The egress QoS class can be set to the same value as the ingress QoS class or to a specific value per CPU extraction queue. A super priority can also be used. For more information, see Super Priorities, page 305.

The mirror field is the mirror probe number, which is set by the analyzer if the frame must be mirrored. There are three mirror probes available, and for each of the probes, associated egress port and egress QoS class can be set by the translation table.

Finally, if the learn-all field is set, the frame should also be forwarded to the stacking links. It is selectable to forward the frame to either of the two stacking links or to both, depending on the stacking topology.

The following table lists the configuration registers associated with the forwarding function.

Table 137 • **Frame Forwarder Registers Overview**

Register	Description	Replication
QFWD::SWITCH_PORT_MODE	Enables forwarding to/from port. Forwarding speed configuration. Drop mode setting.	Per port
QFWD::FRAME_COPY_CFG	Translates CPUQ/mirror/learn all to enqueuing requests.	12
QFWD::FRAME_COPY_LRNA	Learn All frame copy extra port.	None
QFWD::CPUQ_DISCARD	Disables enqueuing to specific CPU queues.	None
QSYS::FWD_PRESSURE	Configures forwarding pressure limits per port.	Per port

Table 137 • **Frame Forwarder Registers Overview *(continued)***

Register	Description	Replication
QFWD::MIRROR_CFG	Configures whether discarded copies should still generate a mirror copy.	None

3.26.4 Congestion Control

The buffer control can only guarantee that no ingress port consumes more than a certain amount of memory. Discards made by the buffer control are tail drops. The congestion control is more advanced in that it facilitates intelligent discard where only frames belonging to a congested flow are discarded.

If the forwarder finds that some frame copies are made to a congested flow, it discards the copies (drop-mode frames) or lets the copies block further processing of the ingress queue (flow control ports).

The congestion control information from the analyzer result involved in the congestion control is listed in the following table.

Table 138 • **Analyzer Congestion Results**

Field	Function
Drop mode	This frame can be dropped even if port is setup for flow control. This function is mainly used by the analyzer for stacking use where the ingress unit's source port determines how congestion should be handled.
DP	Drop precedence level. A value 0-3, telling how yellow the frame is.
QoS class	Frame's ingress QoS class

The current consumption of resources in the congestion controller is updated when a reference is added to one of the egress queues. The congestion controller tracks four different resources:

- Ingress packet memory consumption. Each frame uses a number of 176-byte words.
- Egress packet memory consumption. Each frame uses a number of 176-byte words.
- Ingress frame reference consumption. Each frame uses one frame reference per frame copy.
- Egress frame reference consumption. Each uses one frame reference per frame copy.

There are 5,952 memory words and 5,952 frame references in total.

The accounting is done based on QoS classes and port numbers. Each account has a threshold specifying how many resources the account is permitted to consume. The accounts are as follows:

- Consumption per port per QoS class. This is a reservation account.
- Consumption per port. A frame does not consume resources from this account before the resources belonging to the previous account are consumed. This is a reservation account.
- Consumption per QoS class. A frame does not consume resources from this account before the resources belonging to the previous two accounts are consumed. This is a sharing account.
- Consumption in addition to the above. A frame does not consume from this account before the resources belonging to the previous three accounts are consumed. This is a sharing account.

The congestion controller updates an accounting sheet with the listed accounts for each of the tracked resources.

The following illustration shows reservation accounts for the packet memory accounting sheets when a 700-byte frame from port 0 forwarded to ports 1 and 5 with QoS class 2 is added. The frame size requires the use of five words in the packet memory. The account for ingress port 0, QoS class 2 is reserved three words. Before the frame is added, it is checked if the account was fully utilized. If the account was not fully used, the frame is allowed into the queue system. After the frame is added, the account is updated with the five words and therefore uses two more than was reserved. These are consumed from the port account. The port account is reserved 0 words and the added two words therefore also close the port account. This affects further frame reception.

In the egress side, there are no words reserved for the account per port, per QoS class. The port account for port 1 is reserved 8 words and 0 for port 5.

Figure 90 • **Accounting Sheet Example**

Packet memory, ingress user

Priority	0	1	2	3	4	5	6	7	P
Port									
0			5/3						2/0
1									
…									
…									
…									

Packet memory, egress user

Priority	0	1	2	3	4	5	6	7	P
Port									
0									
1			5/0						5/8
…									
5			5/0						5/0
…									

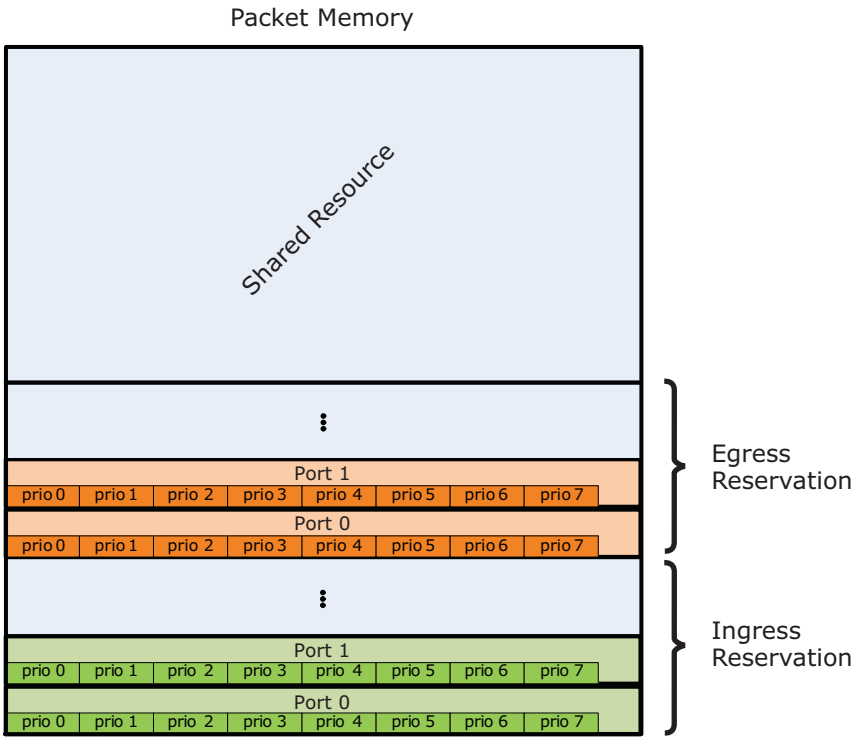
For PFC purposes, the congestion controller contains an additional accounting sheet of the ingress memory consumption. The sheet has its own set of thresholds. The result of the accounting is whether to send PFC pause frames towards the link partner.

The combined rule for allowing a frame copy is if both of the following are true:

- Either ingress or egress memory evaluation must allow the frame copy.
- Either the ingress or egress reference evaluation must allow the frame copy.

When the reserved accounts are closed, the shared accounts spanning multiple ports are checked. There are various rules for regarding these accounts as open or closed. These rules are explained in the following through three major resource modes, which the congestion controller is intended to be used in.

Figure 91 • **Reserved and Shared Resource Overview**



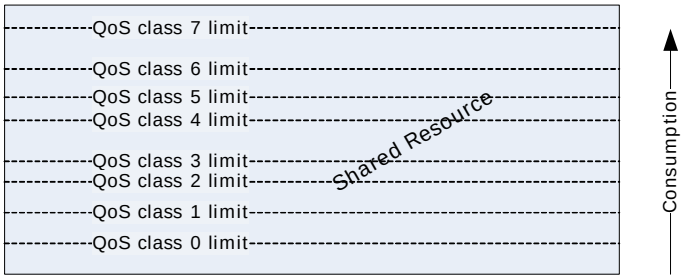
The resource modes are defined by the way the shared resources are distributed between classes of frames. Note, that there are many options for configuring a different resource mode than the modes described in the following. This is outside the scope of this document.

3.26.4.1 Strict Priority Sharing

This resource mode considers higher QoS classes as more important than lower QoS classes. The shared area is one big pool shared between all QoS classes. Eight thresholds define the limit for each of

the eight QoS classes. If the total consumption by all QoS classes exceeds one of the thresholds, the account associated with the particular QoS class is closed.

Figure 92 • **Strict Priority Sharing**



One important property of this mode is that higher QoS classes can block lower QoS classes. A congested flow with a high QoS class uses all of the shared resources up to its threshold setting and frame with lower QoS classes are thus affected (lower burst capacity).

The following table shows the configuration settings for strict priority sharing mode.

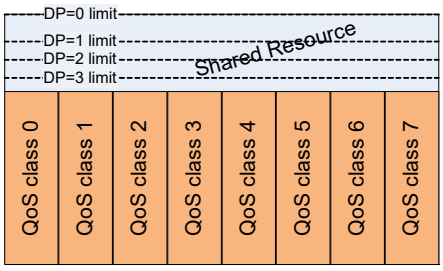
Table 139 • **Strict Priority Sharing Configuration Settings**

Configuration	Setting
Reservation thresholds	These can be set to any desired value. The amount set aside for each account should be subtracted from the overall resource size, before the sharing thresholds are set up.
Ingress QoS class thresholds	Levels at which each QoS class should start to reject further enqueuing. Frames with multiple destinations are only accounted once because they are physically only stored once.
Ingress color thresholds	Set to their maximum value to disable. The color is not considered in this mode.
WRED group memberships	Disable all.
Egress sharing thresholds	Set all to 0, because only use ingress sharing control is . Egress would account for multiple copies, which is not intended.
QoS reservation (QRES::RES_QOS_MODE)	Disable for all. No shared resources are set aside per QoS class.

3.26.4.2 Per Priority Sharing

This mode sets aside some memory for each QoS class and can in addition have an amount of memory reserved for green traffic only. This mode operates on ingress resources, in order to utilize that frames with multiple destinations are only stored once.

Figure 93 • **Per Priority Sharing**



The following table shows the configuration settings for per priority sharing mode.

Table 140 • **Per Priority Sharing Configuration Settings**

Configuration	Setting
Reservation thresholds	These can be set to any desired value, except for ingress per port reservation. The algorithm requires these to be zeroed.
Ingress QoS class thresholds	Set to amount of shared memory which should be set aside for each QoS class. The sum of all these should not exceed the amount of memory left after all the reservations has been subtracted.
Ingress color thresholds	Subtracting all reservations including QoS class shared areas may end up in some unused memory. Set the color thresholds to various levels to guarantee more space the greener.
WRED group memberships	Disable all.
Egress sharing thresholds	Set all to 0, as we only use ingress sharing control. Egress would account for multiple copies, which is not intended.
QoS reservation (QRES::RES_QOS_MODE)	Enable for all.

3.26.4.3 Weighted Random Early Discard (WRED) Sharing

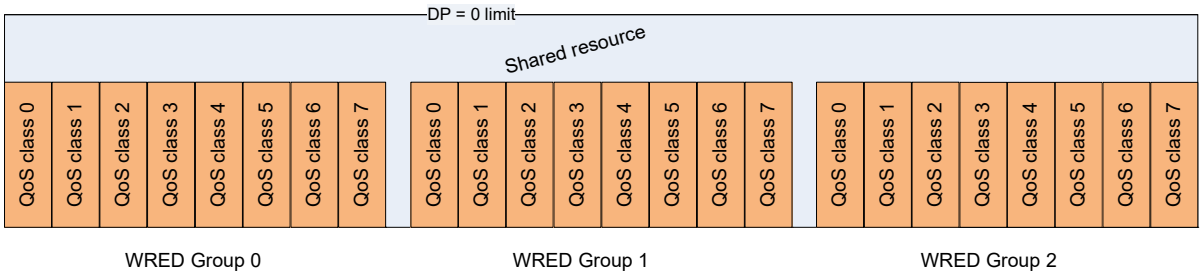
In this mode, the shared resource accounts close at random. A high consumption gives a high probability of discarding a frame. This sharing method only operates on the egress memory consumption.

There are three WRED groups, each having a number of ports assigned to it. Each WRED group contains 24 WRED profiles; one per QoS class and per DP = 1, 2, 3. Each WRED profile defines a threshold for drop probability 0% and a threshold for drop probability 100%. Frames with DP = 0 value are considered green and have no WRED profile. Green frames should always be allowed.

Memory consumptions within a WRED group are tracked per QoS class, but across DP levels, however, green frames affect the drop probability of yellow frames.

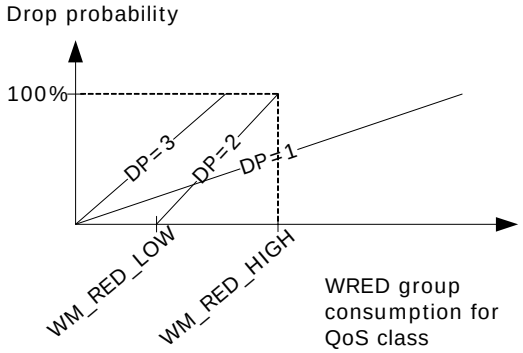
The WRED groups with each eight QoS classes are shown in the following illustration. It is also possible to use fewer groups or different sizes for each QoS classes.

Figure 94 • **WRED Sharing**



The following illustration shows an example of the WRED profiles for a QoS class within a WRED group. Thresholds are shown for DP = 2 only.

Figure 95 • **WRED Profiles**



The WRED sharing is operating as an egress algorithm. The following table shows the configuration setting.

Table 141 • **WRED Sharing Configuration**

Configuration	Setting
Reservation thresholds	These can be set to any desired value, except for ingress per port reservation. The algorithm requires these to be zeroed.
Ingress priority thresholds	Set to their maximum value, as the QoS classes should not have any sharing across WRED groups.
Ingress color thresholds	Set to their maximum.
WRED group memberships	Set to define the WRED groups. For each group, define as well the up to 24 WRED profiles (per QoS class and DP level)
Egress sharing thresholds	Set the QoS class thresholds to the highest possible consumption before any profile hits 100% probability. The color thresholds can then operate on the remainder of the resource in control and can all be set to the same value, allowing use of all the remaining. A discard due to the WRED profile takes precedence over other sharing states, so the area outside the group sections is only available for green traffic.
QoS reservation (QRES::RES_QOS_MODE)	Disable for all.

3.26.4.4 Priority-Based Flow Control

There is a fifth accounting scheme for the pending frames. It operates on ingress memory use, but has the same set of thresholds as the other account systems. The output from that system is however not used for stopping/discarding frame copies, but for requesting the port to transmit pause flow control frames for the involved priorities. The threshold levels here should be configured in a way so that the blocking thresholds cannot be reached during the trailing amount of data.

3.26.4.5 Threshold Configuration

There is a large number of thresholds for the reservations described. They are all found in the QRES::RES_CFG register, which is replicated approximately 5,120 times.

Table 142 • **Threshold Configuration Overview**

QRES::RES_CFG Replication	Description
0 – 119	Ingress memory reservation for port P priority Q accessed at index 8P + Q.
496 – 503	Ingress memory shared priority threshold for priority Q accessed at index 496 + Q.
508 – 511	Ingress memory shared threshold for DP levels 1, 2, 3, and 0.

Table 142 • **Threshold Configuration Overview *(continued)***

QRES::RES_CFG Replication	Description
512 – 526	Ingress memory reservation for port P, shared between priorities, accessed at index 512 + P.
1,024 – 2,047	Ingress reference thresholds. Same organization as for ingress memory, but added offset 1,024.
2,048 – 3,071	Egress memory thresholds. Same organization as for ingress memory, but added offset 2,048.
3,072 – 4,095	Egress reference thresholds. Same organization as for ingress memory, but added offset 3,072.
4,096 – 5,119	PFC memory thresholds, operating on ingress memory use. PFC is activated when no memory left according to these thresholds. Same organization as the ingress stop thresholds, but added offset 4,096.

The following table lists other registers involved in congestion control.

Table 143 • **Congestion Control Registers Overview**

Register	Description
QRES::RES_STAT (replicated same way as RES_CFG)	Returns maximum value the corresponding threshold has been compared with.
QRES::RES_STAT_CUR (replicated as RES_CFG)	Returns current value this threshold is being compared with.
QRES::WRED_PROFILE (72 profiles)	Profile description. There are 3 (grp) × 3 (DP) × 8 (prio) profiles.
QRES::WRED_GROUP	WRED group membership configuration per port.
QRES::PFC_CFG	Enable PFC per port.
QRES::RES_QOS_MODE	Enable QoS reservation for QoS classes.
QFWD::SWITCH_PORT_MODE	Controls whether yellow traffic can use reserved space (YEL_RSVD). Allows ports to use shared space (IGR_NO_SHARING, EGR_NO_SHARING).

3.26.4.6 Frame Forwarder Monitoring

The performance of the frame forwarder can be monitored through the registers listed in the following table.

Table 144 • **Frame Forwarder Monitoring Registers Overview**

Register	Description
QFWD::FWD_DROP_EVENTS	Sticky bits per port telling if drops from each individual ingress port occurred.
QFWD::FWD_CPU_DROP_CNT	Counts number of frames not forwarded to the CPU due to congestion.
QFWD::FWD_STAT_CNT	Counts number of frame copies added to the egress queues in total.
QFWD::FWD_PRESS_DROP_CNT	Counts number of frame copies discarded due to forward pressure. Value is the total discards across all ports.
QSYS::MMGT_IQ_STAT	Returns current ingress queue size.
QSYS::MMGT	Returns number of free references for the egress queues.

3.26.5 Queue Mapping

When a frame is forwarded to an egress port, a frame reference is inserted into one of the port's egress queues. The egress queues are read by a programmable scheduler consisting of a number of scheduler elements (SEs). Each SE serves 8 or 16 queues, configurable through the HSCH::HSCH_LARGE_ENA registers. Each SE selects the next input to transmit and forwards the decision to a next-layer SE, selecting from which of its inputs to transmit. There are a total of three layers, with layer 0 being the queue connected layer. Connections between the layers are fully configurable in HSCH::L0_CFG and HSCH::L1_CFG. For more information about SE functionality, see Scheduling, page 301 and Bandwidth Control, page 302.

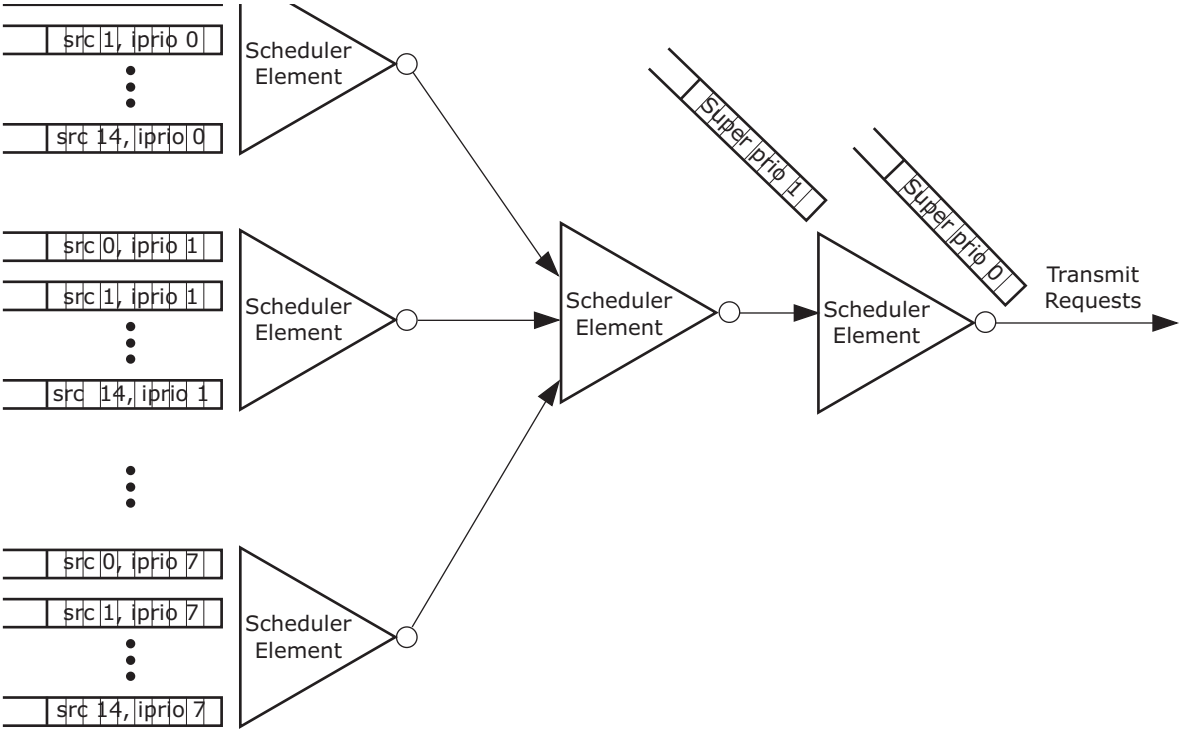
The scheduler includes a dual leaky bucket shaper to limit the traffic rate from inputs attached to it. To distribute the bandwidth between the inputs, the scheduler also contains a weighted round robin arbiter.

The number of queues available in total for all ports is 3,358. The queue mapper must be configured to how these queues are assigned to each egress port. The overall traffic management for each port is the result of the queue mapping and scheduler hierarchy.

The following illustration shows an example of the default egress port hierarchy. At layer 0, eight SEs select between data in the same QoS level, with queues from each source port. This forms a basic traffic manager, where all source ports are treated fair between each other, and higher priorities are preferred. This is accomplished by having the layer 0 SEs select input in a round robin fashion, and the mid-layer select strictly higher inputs before lower inputs. In general, all SEs can be configured to various polices on how to select inputs.

The illustration also shows the two super-priority queues available in all ports. The scheduler hierarchy configuration must be based on a desired scheme. The hardware default setup is providing a queue per (source, priority) per egress port. All queues with the same QoS level are arbitrated first, in layer 0.

Figure 96 • **Scheduler Hierarchy (Normal Scheduling Mode)**

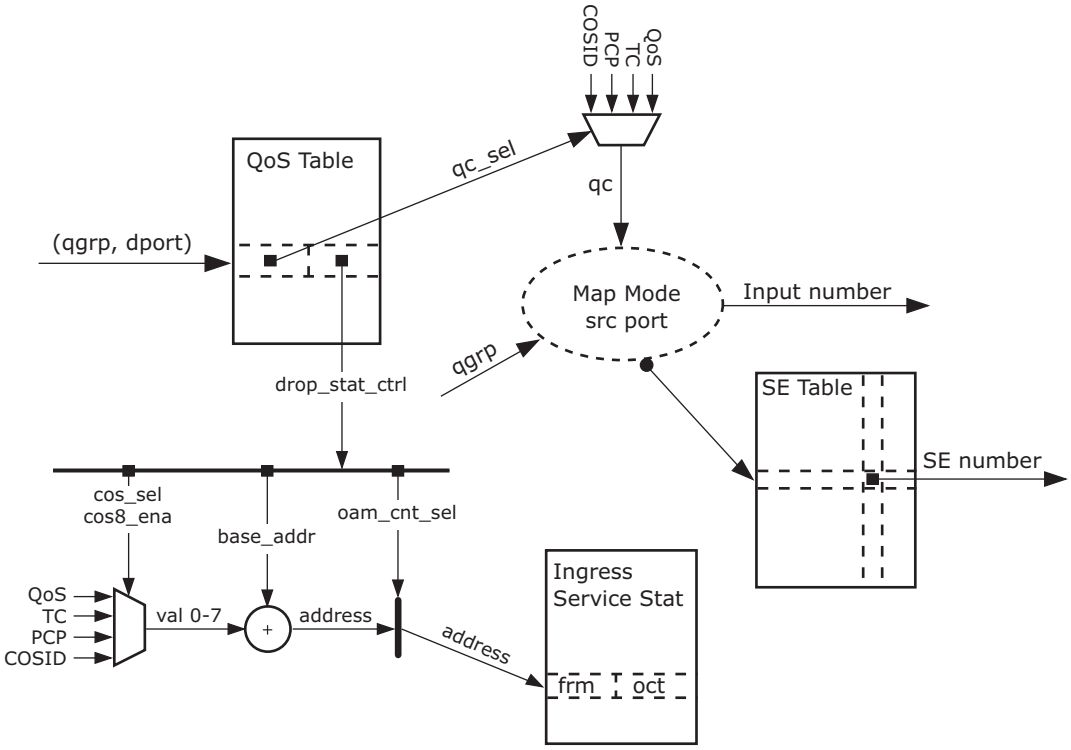


3.26.5.1 Mapping System

All queues in the system are referred to as a layer 0 SE number and an input number. For example, the default configuration puts all traffic for port 0 from source 7 in priority 0 into the queue (0,7); SE number 0, input 7. When the forwarder requests a frame to be added to a queue, the queue group—all classified priorities, the source port, and the destination port—are used to find the SE and input on it, through some

configurable mapping tables. There are three basic hierarchy types, and the first part of the mapping procedure is to find the hierarchy type in QFWD::QMAP_PORT_MODE. The mode is found for frames with classified qgrp = 0 in the QMAP_MODE_NONSERVICE field and in QMAP_MODE_SERVICE for qgrp >0. Note that service and non-service mappings can be combined at the higher layers, in order to combine the two basic types of traffic in the application. An overview of the queue mapping tables is shown in the following illustration. The drop statistics mode is also found by looking up information in these tables. For more information, see Statistics, page 306.

Figure 97 • **Queue Mapping Tables**



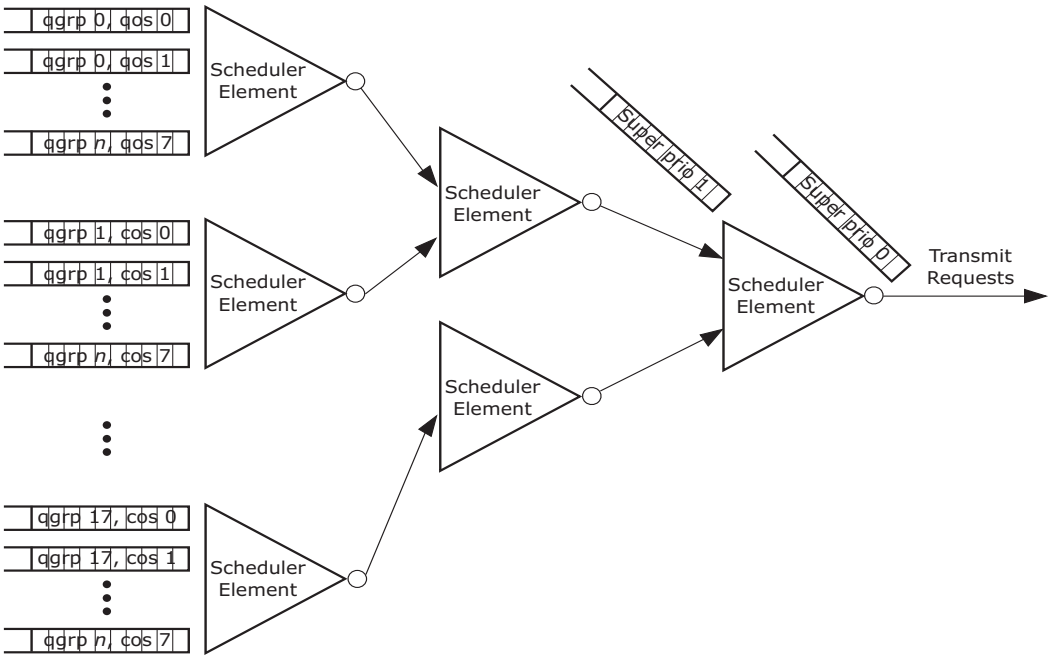
Mapping is accomplished by the following steps.

1. Find queuing class (qc_cl)
   qc_sel = QMAP_QOS_TBL (qgrp,dport)
   qc = (qos, cosid, pcp, tc) (qos_sel)
2. Find queue. This mapping depends on which of the following hierarchy modes are configured:
   Mode 0: Normal mode. This is the default mode on all ports. It is used for arbitrating all frames with the same queuing class, treating all source ports equally. SE = QMAP_SE_TBL (128 + qc, dport), inp = source port.

   Mode 1: Group mode. This mode is used for having eight queues per queue group in layer 0. Used in Carrier applications where each service should have its own queues, with a traffic profile per queuing class.
   SE = QMAP_SE_TBL (qgrp,dstp), inp = qc.

The following illustration shows the group scheduling mode.

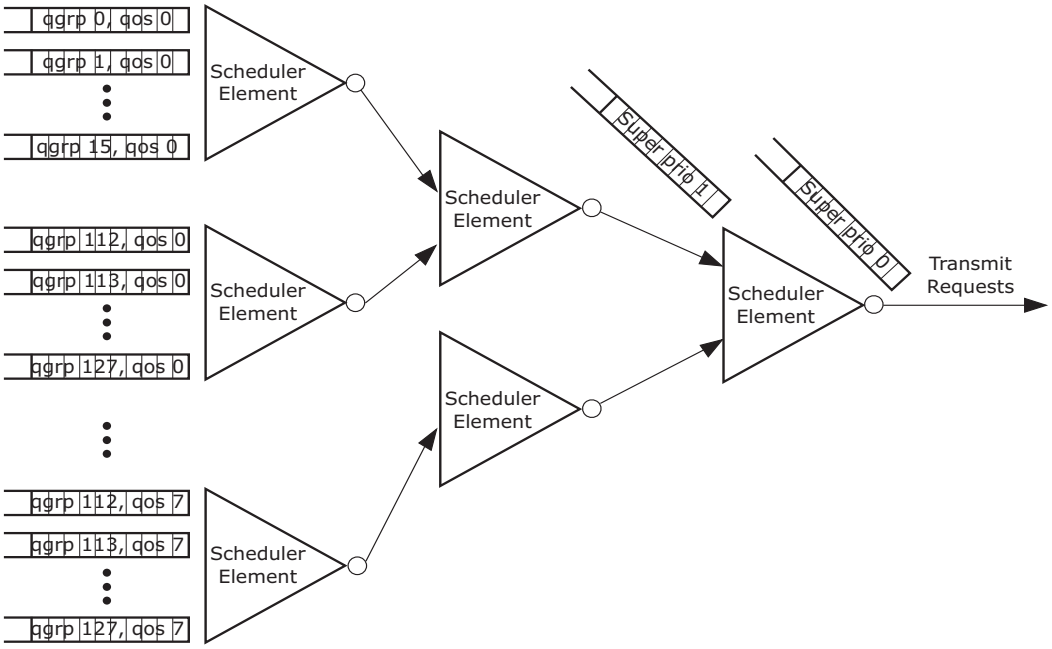
Figure 98 • **Group Scheduling Mode**



Mode 2: Microwave Backhaul (MBH) mode. This mode is used when many queue groups need to be arbitrated per queuing class, and they share a common outgoing quality of service policy. For example, it may be possible to have 100 queues per queuing class arbitrated on level 0 and 1, and class selected on level 2. In this case, the SE table is used by SE = QMAP_SE_TBL (qgrp – (qgrp MOD 16) + qc, dport), inp = qgrp MOD 16.

The following illustration shows the Mobile Backhaul mode.

Figure 99 • **Mobile Backhaul Mode**



The following table lists the registers associated with assigning queues for each egress port.

Table 145 • **Queue Mapping Registers Overview**

Configuration	Description
QMAP_PORT_MODE	Selects queue layer hierarchy type
QMAP_QOS_TBL	Selects per (qgrp,dport) from which queueing QoS it should be derived
QMAP_SE_TBL	Selects per (qgrp,dport) a scheduling element

3.26.5.2 Service Statistics

As shown in the following illustration, drop statistics configuration is also being looked up in the mapping flow. The device has 1,024 service counter sets. On the ingress side, the queue system only counts discarded frames. On the egress side, it counts the amount of frames transmitted. In both cases, there is a counter for green frames and another counter for yellow frames.

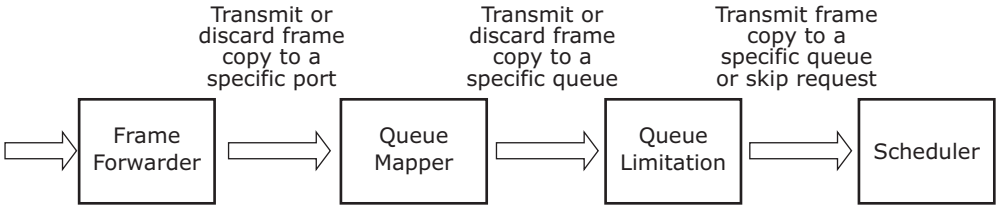
The egress counter set to count is selected through the rewriter. The ingress counter is selected by looking up drop statistics information when the queuing class is looked up. That returns a base index, which is multiplied by four, and one of the four classified priority parameters is added. If the set is configured for using only four counters, the MSB of the priority is cleared. The resulting index will afterwards be counting drops, with 18 bits of octet counting, and 10 bits of frame counting. Optionally, all 28 bits can be used for frame counting. For more information about drop statistics, see Statistics, page 306.

3.26.6 Queue Congestion Control

The congestion control system may choose to forward a frame to a specific queue or to discard it. In either case, the decision passes through a queue limitation function, which can alter the decision based on another view of the resources.

The following illustration shows the drop decision flow.

Figure 100 • **Drop Decision Flow**

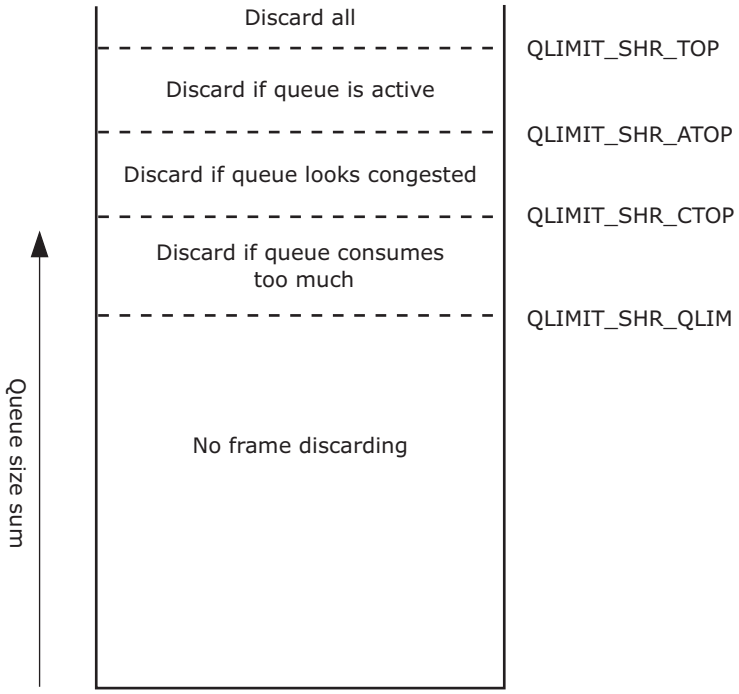


The queue limitation function checks the queue size against various configurable and dynamic values and may change a transmit request to a discard by canceling the request to the scheduler.

The queue limitation system uses queue sizes (in terms of packet memory usage) of all queues. Each queue is assigned to a share. The share is a configuration parameter for the port, QoS class, and service/non-service type as set in the QLIMIT_PORT_CFG register. With this configuration granularity, it also possible to change the queue rights to be shared for the scheduling element to which they are attached, so that one aggressive queue will make other queues on the same element start discarding frames. That is done in the QLIMIT_SHR_MODE field. A set of watermarks is afterwards used to control some queue limitation drop mechanisms, trying to discard data for congested queues without affecting well-balanced traffic.

The conditions under which a frame is discarded is shown in the following illustration The watermarks shown are defined per shared account and per color. If yellow traffic is present, it can be discarded fairly without affecting the green traffic.

Figure 101 **• Queue Limitation Share**



The following table provides descriptions of the queue limitation conditions.

Table 146 **• Queue Limitation Conditions**

State	Description	Traffic Condition
Queue is active	The queue size exceeds the watermark in QLIMIT_QUE_ACT	The queue is not idle. When the filling gets very high, only non-active queues are allowed further enqueing.
Queue looks congested	The queue size exceeds the watermark in QLIMIT_QUE_CONG	The queue seems to be growing. When the share filling is high, it can be chosen to discard further data into the queue.
Queue consumes too much	The queue size exceeds DYN_WM, which is the QLIMIT_SHR_CTOP watermark divided by the number of queues looking congested.	The queue seems to be one of the reasons for the growing memory consumption, and further additions to the queue can be avoided.

The following table lists the queue limitation registers.

Table 147 **• Queue Limitation Registers**

Register Groups	Description
QLIMIT_QUEUE	Reads current queue size
QLIMIT_PORT	Assigns ports to shares
QLIMIT_SHR	Configures watermarks for share and monitor filling and dynamic watermarks
QLIMIT_MON	Reads the status for each of the logical shares.

3.26.7 Scheduling

All egress queues are combined in a tree structure where the root is connected to an egress port. The scheduling capabilities consist of the following three aspects.

- Structure of the hierarchy
- Bandwidth limitation (shaping)
- Bandwidth distribution (arbitration between inputs)

3.26.7.1 Hierarchy Structure

The hierarchy consists of three layers: queue, middle, and outer.

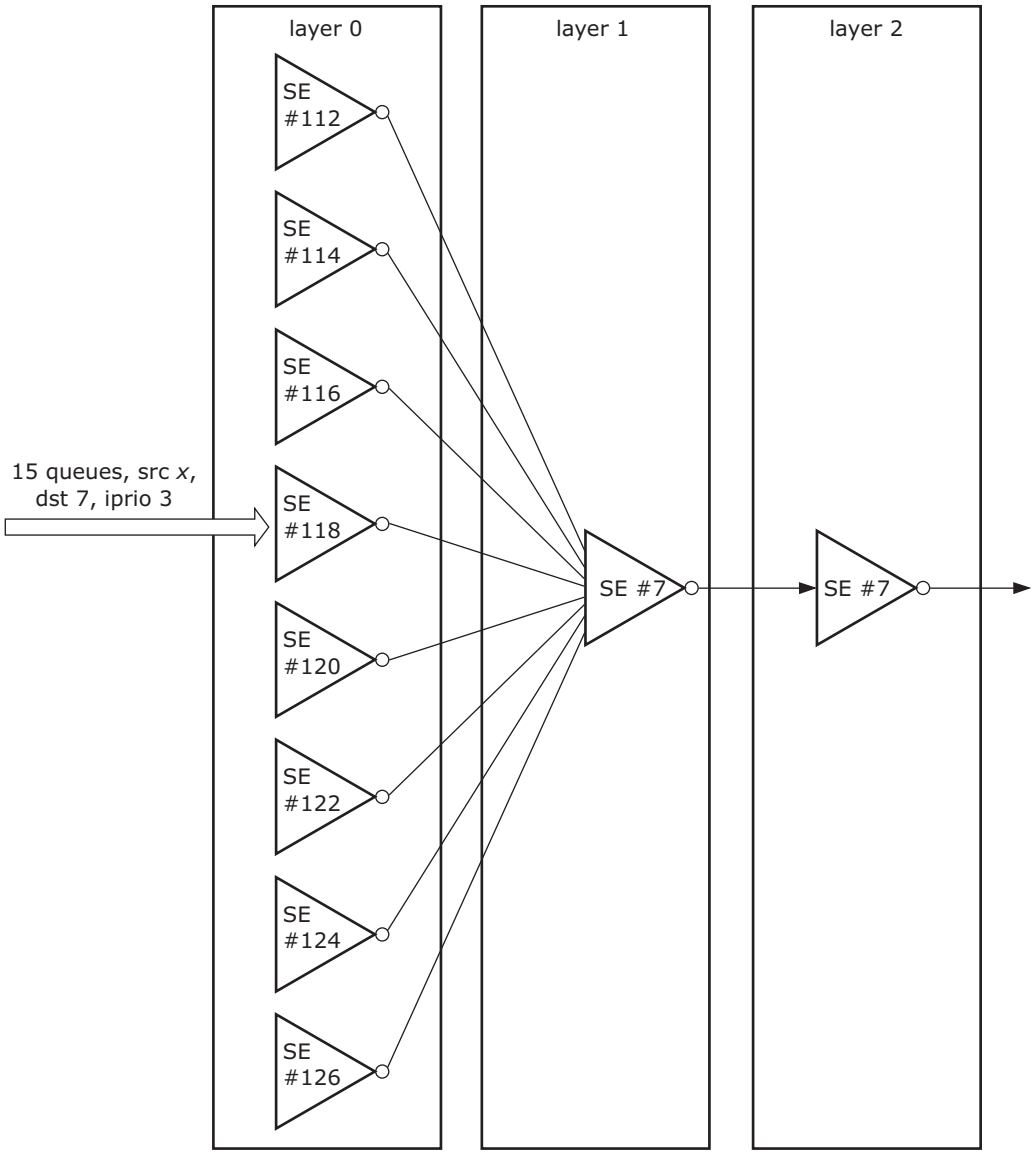
- Layer 0 is the queue level. All inputs to scheduler elements are connected to a specific egress queue, as described in the previous section. This layer has 416 elements with eight inputs each. A number of those can be reconfigured to have 16 inputs, in which case they are called “large elements”. Only elements in multiples of two can become large, so the one following cannot be used.
- Layer 1 is the middle layer. There are 64 elements in this layer, all having 32 inputs. These inputs are connected to outputs of the layer 0 elements, through the HSCH::L0_CFG table.
- Layer 2 is the output layer. There is one scheduler element per egress port (15 elements). All elements have 64 inputs connected to outputs from layer 1 elements. Input *n* on the layer 2 elements can only be connected to layer 1 element *n*, configured in HSCH::L1_CFG.

3.26.7.2 Default Hierarchy

The initialization engine in the scheduler builds a default hierarchy for all ports, and no further configuration of it is needed. It sets all ports into normal mode, using eight large elements on layer 0, and one element on each of the two other layers. The layer 0 elements are all configured for frame based round robin selection, and the layer 1 elements for strict selection of highest available input.

Port *n* uses elements 16*n*, 16*n* + 2, 16*n* + 14,... for layer 0. The example in the following illustration is for port 7.

Figure 102 • **Default Scheduling Hierarchy**



3.26.7.3 Bandwidth Control

Each scheduler element has a dual leaky bucket shaper on its output, which may block further transmissions from that scheduler element. It is controlled through a leaky bucket system, filling the bucket when data is transmitted, and leaking it at the configured rate. The bucket is closed when more data than the configured burst capacity has been filled into it.

The shaper has three rate modes that define what is added to a bucket:

- Bits transmitted excluding IFG
- Bits transmitted including IFG
- Frames transmitted

Each of the three layers have four linked lists configured. All scheduling elements with active shapers must be members of one of those. Each linked list is processed once every configurable interval. When a list is processed, all shapers in it will leak its configured rate setting.

Examples:

- Data rate shapers are connected in a linked list, which is traversed every 10 µs. The shaper rate granularity becomes 100 kbps.
- Frame rate shapers are connected in a linked list, which is traversed every 1 ms. The shaper rate granularity becomes 1 kilo frames per second.

Each scheduler element includes two leaky buckets to support DLB shaping. The committed information rate (CIR) bucket is open when the filling is below the configured limit. The excess information rate (EIR) shaper is always closed, unless the congestion control reports that a threshold is reached. Which congestion thresholds to react on are configurable for each shaper and should depend on the traffic groups the particular element is a part of. The effect of the DLB shaper is that the rate out of the shaper is CIR or CIR plus EIR, depending of the accumulation of data in the queue system.

The following table lists the registers associated with shaping.

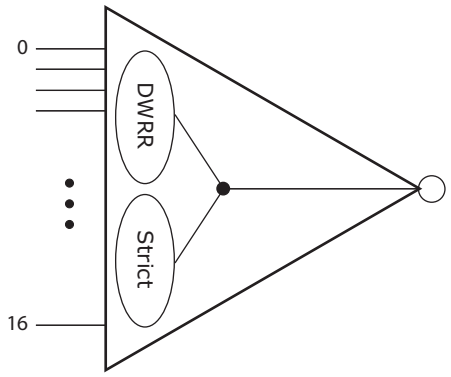
Table 148 • **Shaper Registers Overview**

Register	Description
HSCH::SE_CFG	Selects filling mode
HSCH::CIR_CFG	Configures leak rate and burst capacity for CIR bucket
HSCH::EIR_CFG	Configures leak rate and burst capacity for EIR bucket
HSCH::SE_DLB_SENSE	Selects thresholds to control EIR with.
HSCH::HSCH_LEAK_LIST	Configures leak list periods and starting index
HSCH::SE_CONNECT	Links to next in leaking lists
HSCH::CIR_STATE	Returns current CIR bucket filling
HSCH::EIR_STATE	Returns current EIR bucket filling
HSCH::SE_STATE	Returns whether or not the shaper is open

3.26.7.4 Bandwidth Distribution

Each scheduler element is selecting one of its inputs as the next source of transmission if a scheduling process passes through it. A number of different options are available for how this arbitration takes place. A configured value for the scheduler element tells how many of the lower indexed inputs should be arbitrated with the DWRR method. Inputs with higher indexes are preferred and arbitrated strict.

Figure 103 • **Scheduler**



The lower indexes are arbitrated in the effort to achieve a bandwidth ratio between them. In this regard, the bandwidth can be measured in frames, data bits, or line bits (including IFG). Each input is configured with a cost setting, and the algorithm strives to give *n* times more bandwidth to input A than B, if the cost configuration for A is *n* times lower than B. The following are two cost configuration examples.

- Input 4 has cost 7, and input 6 has cost 11. If the total bandwidth shaped out of the scheduler element is 180 Mbps, input 4 will get 110 Mbps of it.
- All inputs have cost 1, and the element is configured for frames per second arbitration. Algorithm will by this strive to send an equal number of frames from each input.

Configuring costs is done indirectly by setting the HSCH_CFG_CFG.DWRR_IDX first (pointing to the scheduler element to configure), after which the costs are accessible through the HSCH_DWRR registers.

The following table lists the registers associated with the arbitration algorithm.

Table 149 • **Scheduler Arbitration Registers Overview**

Register	Description
HSCH::SE_CFG	Selects arbitration mode (data/line/frame) and number of inputs running DWRR
HSCH::CFG_CFG	Selects for which element to configure/monitor DWRR algorithm
HSCH::DWRR_ENTRY	Configures costs for each input
HSCH::INP_STATE	Returns input states for a scheduler element

3.26.7.5 Queue Shapers

In addition to the shaping capabilities at the output of each scheduling element, it is also possible to shape individual queues. These shapers are single leaky bucket based. The devices contain 832 such shapers that can be assigned to any of the 3,328 queues. The assignment of queue shapers is done through the HSCH::QSHP_ALLOC_CFG register. This is replicated per scheduling elements in layer 0, and defines the range of inputs on the element that should have queue shapers assigned, and the index of the first of the 832 shapers to allocate to it. The shapers must be linked together for the leaking process in HSCH::QSHP_CONNECT

For example, to configure queue shapers on inputs 5 to 7 on elements 100 to 120, use the queue shapers from 400 to 462.

```
For I in 100 to 120 loop
   HSCH:QSHP_ALLOC_CFG[I]:QSHP_ALLOC_CFG.QSHP_MIN := 5
   HSCH:QSHP_ALLOC_CFG[I]:QSHP_ALLOC_CFG.QSHP_MAX := 7
   HSCH:QSHP_ALLOC_CFG[I]:QSHP_ALLOC_CFG.QSHP_BASE := 3*(I-100)+400
   HSCH:QSHP_ALLOC_CFG[I]:QSHP_CONNECT.SE_LEAK_LINK := I+1;
Endloop
(terminate chain)
HSCH:QSHP_ALLOC_CFG[120]:QSHP_CONNECT.SE_LEAK_LINK := 120
```

The leak process operates as for the SE shapers and must be configured in HSCH:HSCH_LEAK_LISTS[3].

The indexes of shapers allocated to a set of queues are only used in the QSHP_BASE settings. The shapers are accessed afterwards through the QSHP_CFG and QSHP_STATUS registers, accessed indirectly after setting HSCH_CFG_CFG.CFG_SE_IDX to the desired scheduling element.

For example, to configure the shaper on input 6 of element 118 to shape to 1.2 Mbps, set HSCH::HSCH_CFG_CFG.CFG_SE_IDX := 118 and HSCH:QSHP_CFG[6].QSHP_CIR_CFG.CIR_RATE := 12 (assuming the leak chain is configured for unit 100 kbps).

3.26.7.6 Miscellaneous Scheduler Features

The following table lists some features that might be useful.

Table 150 • **Miscellaneous Scheduler Registers Overview**

Field	Description
SE_CONNECT_VLD	Stops the whole scheduler operation. Can be useful when reconfiguring the hierarchy.
LEAK_DIS	Disables leaking process.
FRM_ADJ	Sets the byte difference between line and data rate calculations.
DEQUEUE_DIS	Disables transmissions per port.

3.26.8 Queue System Initialization

The queue system automatically initializes all tables and data structures when setting RAM_CTRL.RAM_ENA to 1, followed by setting RAM_CTRL.RAM_INIT to 1. Software should afterwards wait for 150 µs or wait for the RAM_INIT bit to be cleared by hardware. After that the RESET_CFG.CORE_ENA must be raised, and the queue system is ready to operate. All thresholds and queue mappings are prepared with operational values. Configuration can afterwards be modified. The only per-port setting required to change in the queue system for initial bring-up of the queue system is the SWITCH_PORT_MODE.PORT_ENA, which must be set to one allowing switching of frames to and from the port.

3.26.9 Miscellaneous Features

This section provides information about other miscellaneous features of the queue system.

3.26.9.1 Frame Aging

The queue system supports discarding of frames pending in the queue system for too long time. This is known as frame aging. The frame aging period is configured in FRM_AGING.MAX_AGE, which controls the generation of a periodical aging tick.

The packet memory manager marks frames pending in the queue system when the aging tick occurs. Frames pending for more than two aging ticks are subject to frame aging and are discarded by the queue system when scheduled for transmission. As a consequence, frames can be discarded after pending between one or two aging periods in the queue system.

The following table lists the configuration registers associated with frame aging.

Table 151 • **Frame Aging Registers Overview**

Field	Description	Replication
QSYS::FRM_AGING.MAX_AGE	Sets the aging period	None
HSCH::PORT_MODE.AGE_DIS	Disables aging checks per egress port	Per port
HSCH::PORT_MODE.FLUSH_ENA	Regards various frames as being too old	None

3.26.9.2 Super Priorities

There are two special queues available in each egress port: super priority queue 0 and super priority queue 1. For more information, see Figure 96, page 296. These queues are selected before any other queues. They are intended for urgent protocol traffic injected by the CPU system. Frames in super priority queue 0 disregard the output shaper state and are not counted in the shaper's buckets. Frames in super priority queue 1 obey the output shaper state and are counted in the shaper's buckets.

The following frames can be added to the super priority queues:

- AFI injected frames: The AFI can be configured to inject into either of the super priority queues. For more information, see Adding Injection Frame, page 320.
- CPU injected frames: Frames injected with an IFH with the SP flag set are inserted into the super priority queue as being set in the drop precedence (DP) IFH field.
- CPU extracted, mirror, learn all frames: Frames subject to the translation in QFWD::FRAME_COPY_CFG can configure use of either of the super priority queues.

3.26.9.3 Frame Modifications in the Queue System

The queue system typically passes data completely untouched from the ingress to the egress side, except for a few special cases.

Frame copies to the CPU can be sent into the rewriter with two different priority schemes, involving modification of the internal frame header. Either the frame takes the QoS class from the CPU extraction queue number or it takes the QoS class from the configuration in QFWD::FRAME_COPY_CFG. This is controlled in PORT_MODE.CPU_PRIO_MODE.

Frame copies due to learn-all stack synchronization can be truncated to 64 bytes to save bandwidth on the line. This is enabled in HSCH::PORT_MODE.TRUNC_ENA.

3.26.9.4 Statistics

The queue system counts frame discards and successful transmissions. Each counted event updates a counter instance, counting octets in a 40-bit counter and frames in a 32-bit counter. A frame is counted as discarded when none of the frame's destination ports are reached, excluding mirror and learn-all copies. If a frame is discarded, a counter instance, per ingress port, per QoS class per color, is updated. The color granularity here is only green or yellow, mapped from the analyzed DP value through the QSYS::DP_MAP settings. A frame transmission is a frame leaving the queue system towards the rewriter. In this direction, a counter instance per egress port, per QoS class per color, is updated. An abort counter instance per egress port counts frames discarded due to frame aging, queue system flushing, or abort requests from the rewriter.

There are statistics available per service counter index, which is provided by the analyzer for ingress service discard statistics and from the rewriter for egress service transmission statistics. The service counters are available per counter index, per color.

Access to the statistics is indirect. The QSYS::STAT_CFG configuration register must be set to the port or service counter index to be accessed, and all counters related to this index can afterwards be read in the QSYS::CNT registers. The following table lists the available port counters and how they are mapped. Unused addresses return undefined values. In the expressions, yel = 1 for reading yellow counters and yel = 0 for green. The mapping from classified DP to color is made in the QSYS::DP_MAP register.

The following table shows the addresses of the frame counters. The equivalent octet counters are found at the frame counter address plus 64 for the 32 least significant bits and at the frame counter address plus 128 for the eight most significant bits.

Table 152 • **Statistics Address Mapping**

Statistics Index	Event
0 – 15	Frames received on the port viewed. The first eight counters are for green frames, QoS level 0 – 7. The last eight counters are for yellow frames, defined by the QSYS::DP_MAP register.
16 – 31	Frames dropped on the port viewed. Drops will be for the ingress port point of view or egress, depending on the DROP_COUNT_EGRESS option. There are eight green counters and eight yellow counters.
32	Tail drops on the port views. Drops made by the buffer control.
256 – 271	Transmitted frames on the port viewed. Same layout as for the receive set.
272	Transmit aborted frames for the port viewed. This can be due to aging, flushing, or abortion decision by the rewriter.
512 + 513	Green and yellow drops for service counter viewed.
768 – 769	Green and yellow transmissions for service counter viewed.

The service counters are reduced in size; 10 bits for the frame part and 18 bits for the octet part. However, STAT_CFG.STAT_SRV_PKT_ONLY can be set to use all 28 bits for frame counting.

The following table lists the registers associated with statistics access.

Table 153 • **Statistics Registers Overview**

Register	Description	Replication
QSYS::STAT_CFG	Selects which counter set index to access. Clears counters.	None
QSYS::CNT	Replicated register with statistics access.	1,024
QSYS::DP_MAP	Maps DP levels to color.	None

Examples:

To read number of transmitted yellow frames for service counter 714:

1. Write 714 to STAT_VIEW in QSYS::STAT_CFG.
2. Read QSYS::CNT[385].

To read number of discarded green octets with QoS class 4 on port 9:

1. Write 9 to STAT_VIEW in QSYS::STAT_CFG.
2. Read QSYS::CNT[76] for the least significant bits.
3. Read QSYS::CNT[140] for the most significant bits.

3.26.9.5 Energy Efficient Ethernet (EEE) Control

The front ports are able to signal low power idle (LPI) towards the MACs, in case they request the PHYs to enter low power mode. A controller in each port module assures that frames are not transmitted when the PHYs are powered down. This controller powers the PHY down when there is nothing to transmit for a while, and it powers up the PHY when there is either data ready for a configurable time or when the queue system has urgent data ready requiring transmission as soon as possible.

The following table lists the registers with configuring EEE.

Table 154 • **EEE Registers Overview**

Register	Description	Replication
QSYS::EEE_CFG	Configures the priorities per port which should bring the port alive as fast as possible.	Per port
QSYS::EEE_THRES	Configures the amount of cells or frames pending for a priority before the port should request immediate power up.	Per port

Example: Configure the device to power up the PHYs when 10 frames are pending.

Only ports 3 through 6 should regard their data as urgent and only for priorities 6 and 7. All other data should get transmitted when the EEE controller in the port has seen pending data for the controller configured time.

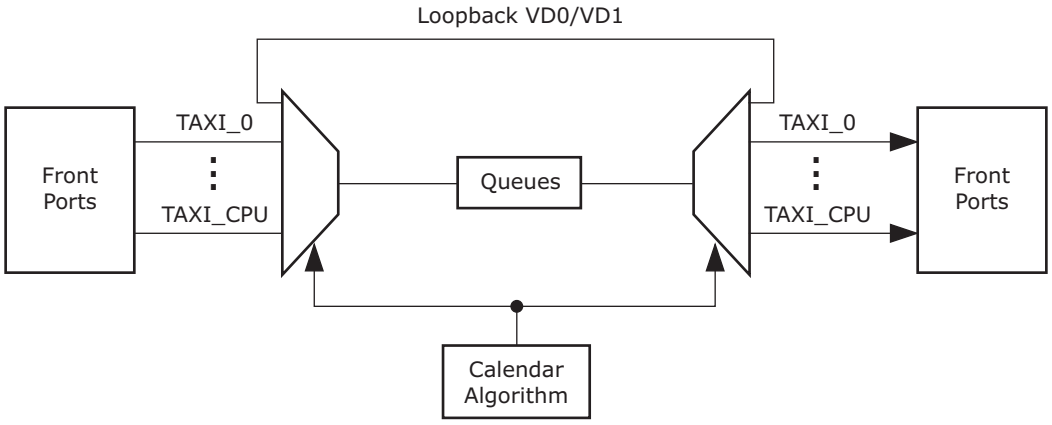
1. Write 10 to QSYS::EEE_THRES:EEE_HIGH_FRAMES.
2. Write 0xC0 to QSYS::EEE_CFG[3–6]::EEE_FAST_QUEUES.

3.26.9.6 Calendar Setup

There are a number of logical ports, all needing bandwidth into and out of the queue system. This is controlled through a calendar module that grants each port access to internal busses.

The following illustration shows an overview of the internal busses. All ports are connected to a port data bus (taxi bus) and granted access to the chip core through a mux obeying the calendar decision. Similarly, on the path towards the transmit interface, a mux controls which port transmits the next time.

Figure 104 • **Internal Bus**



The front ports are connected to taxi busses and must be guaranteed bandwidth that corresponds to their line speed. The internal CPU ports and loopback paths can operate up to 10 Gbps (7.5 Gbps for VD0).

In any port configuration, a subset of the ports is enabled. The calendar algorithm is the central place where bandwidth is distributed between the ports. The calendar algorithm can be controlled by two methods: automatic or sequenced.

In the automatic calendar configuration, a configuration per port defines if the port is granted 0 Gbps, 1 Gbps, 2.5 Gbps, or 10 Gbps bandwidth. For the four internal ports, the bandwidth granted is half of these choices. In other words, internal CPU port 1 can be granted 1.25 Gbps in bandwidth.

In the sequenced operation, a specific sequence of port numbers forms the calendar. The port sequence controls which port is granted the cycle, per bus cycle (two system clock periods).

The algorithm mode is configured in CAL_CTRL.CAL_MODE. It is possible through this register to program a manual sequence, to halt the calendar, and to select between sequenced and automatic calendars.

Both calendar modes may leave some slots unallocated and therefore idle on the busses. These idle cycles can be used for configuration access (which will wait for unused cycles) and other slow operations in the system (MAC table scan, for example). Slots allocated for a port and not used due to lack of data to transfer can be used by the internal ports through the HSCH::OUTB_* registers, where it is, per internal port, configured how often it will seek granting of an unused cycle. Setting internal CPU port 0 to seek access every five cycles means that it can get up to one frame transfer every 64 ns, or approximately 10 Gbps.

In the ingress side, the loopback paths can use idle cycles if enabled in the assembler, which controls the ingress mux.

The following table list the registers associated with configuring the calendar function.

Table 155 • **Calendar Registers Overview**

Register	Description
CAL_CTRL	Calendar function mode. Configures auto-grant rate.
OUTB_SHARE_ENA	Enables granting unused bus cycles to the internal ports (CPU and VD) on the egress side.
OUTB_CPU_SHARE_ENA	Configures bandwidth sharing between the two internal CPU ports.
CAL_SEQ	Interface to the sequence memory. See register list.
CAL_AUTO	Per port setting with requested bus bandwidth.

3.27 Automatic Frame Injector

The automatic frame injector (AFI) provides mechanisms for periodic injection of PDUs such as the following:

- OAM PDUs for continuity check, loss and delay measurement
- OAM PDUs for high service activation test (ITU Y.1564)
- IEEE 1588 PDUs

The following blocks are involved in frame injection.

- QSYS. The frames are sent into the queue system by the CPU and stored in the queue system until deleted by software.
- AFI. Using timers and other parameters, AFI controls the automatic frame injection.
- REW/OAM_MEP. For outbound data path injections, the REW and OAM_MEP performs modifications of OAM frames.
- Loopback path. The loopback path (VD1) between the disassembler and assembler is used to inject frames into the inbound data path.
- ANA/OAM_MEP. For inbound data path injections, the ANA and OAM_MEP performs modifications of OAM frames.

AFI supports two types of automatic frame injection: timer triggered (TTI) and delay triggered (DTI).

- Timer Triggered Injection (TTI). Frame injection rate is controlled by timers with typical values of approximately 3 ms or higher. Only single-frame injections are supported; injection of sequences of frames is not supported.

  For each frame, jitter can be enabled for the associated injection timer.

  Timer triggered injection can be combined with delay triggered injection.
- Delay Triggered Injection (DTI). A sequence of frames, including configurable delay between frames, is injected.

  Delay triggered injection can be combined with TTI, such that DTI is used to inject a (high) background load (imitating normal user traffic), and TTI is used to inject OAM PDUs for measuring IR, FLR, FTD, and FDV.

  AFI supports up to 16 active DTI frame sequences at a time.

3.27.1 Injection Tables

The main tables used to configure frame injection are frame table, DTI table, and TTI table.

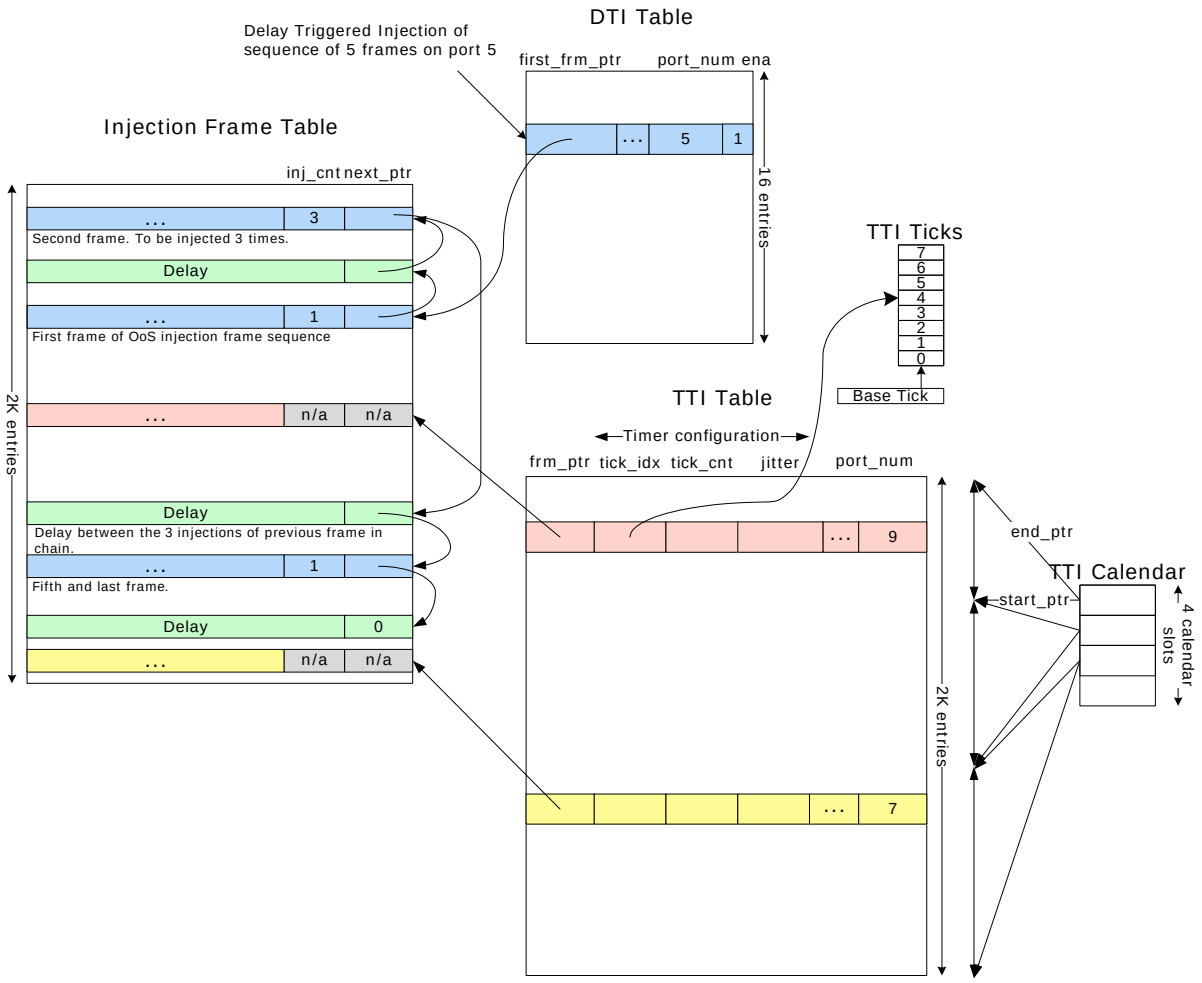
Injection frames for both TTI and DTI are configured in the frame table. The frame table itself does not hold the frame, but rather a pointer to where the frame is located in the buffer memory within QSYS.

For DTI, frames can be configured into a linked list of frames to be injected. Each frame in the linked list may be configured to be injected multiple times before proceeding with the next frame in the chain. Each frame in the sequence is followed by one or more delay entries, specifying the delay between injections.

Timers for each TTI frame are configured in the TTI table. TTIs are always single frame injections, so the inj_cnt and next_ptr fields in the frame table are unused for TTI. Delay entries are not used for TTI.

The TTI calendar controls the frequency with which different parts of the TTI table are serviced.

Figure 105 • **Injection Tables**



3.27.2 Frame Table

The following table lists the registers associated with configuring parameters for each entry in the frame table.

Table 156 • **Frame Table Entry Register Overview (2048 entries)**

Register	Field	Description
FRM_NEXT_AND_TYPE	NEXT_PTR	Next entry in table. Does not apply for TTI.
FRM_NEXT_AND_TYPE	ENTRY_TYPE	Entry type: Frame or Delay entry. Does not apply for TTI.
ENTRY_TYPE=Frame FRM_ENTRY_PART0	INJ_CNT	Number of times to inject frame. Does not apply for TTI.
	FRM_RM	Frame removal. See Removing Injection Frames, page 321.
	FRM_GONE	Frame removal.
	FRM_INFO	Frame information. See MISC:NEW_FRM_INFO.FRM_INFO.
ENTRY_TYPE=Delay FRM_ENTRY_PART0	DELAY	Delay between injection of start of frames. Unit is one system clock cycle. Does not apply for TTI.

3.27.3 Delay Triggered Injection

Delay triggered injection supports up to 16 flows at a time. For each DTI flow, the DTI table holds a pointer to a sequence of frames and delay entries, as well as the queue and port number, into which the frames are injected. The NEXT_PTR field of the frame table is used to link frames and delay entries together into a sequence. The frame sequence can have any length, only limited by the size of the frame table.

Each DTI sequence can be configured to be injected repeatedly until stopped, or injected a specified number of times. For example, if the frame sequence consists of five frames with INJ_CNT = 1, and the sequence is configured to be injected 1000 times, then a total of 5000 frames will be injected.

DTI can be used in conjunction with TTI, for example, using TTI to inject OAM PDUs for measuring performance metrics of the service.

The following table lists the registers associated with configuring the parameters for each of the 16 entries in the DTI table.

Table 157 • **DTI Table Entry Registers Overview**

Register	Field	Description	Replication
DTI_MODE	MODE	Mode of DTI instance.	Per DTI
DTI_MODE	TRAILING_DELAY_SEQ_CNT	Only applies “trailing delay” for every Nth sequence injection. See Fine Tuning Bandwidth of Multiframe Sequence, page 312.	Per DTI
DTI_MODE	DTI_NEXT	Used for concatenating DTIs. Next DTI to activate when this DTI completes.	Per DTI
DTI_FRM	FIRST_FRM_PTR	Pointer to first frame in frame sequence.	Per DTI
DTI_FRM	NEXT_FRM_PTR	Pointer to next frame in frame sequence.	Per DTI
DTI_CNT	CNT	Counter (mode dependent).	Per DTI
DTI_PORT_QU	PORT_NUM	Port number on which injection queue transmits.	Per DTI

Miscellaneous DTI related parameters are shown in the following table.

Table 158 • **Miscellaneous DTI Parameters**

Register	Field	Description	Replication
DTI_CTRL	BW	Bandwidth of DTI flow 0: <5 Gbps 1: >=5 Gbps	Per DTI
DTI_CTRL	ENA	Enables DTI	Per DTI

Frame entries in a DTI sequence may point to the same frame in the queue system’s buffer memory. For example, shown in the following illustration, frame A and frame B shown may point to the same frame in the buffer memory.

The delay entries in a DTI sequence must be configured such that they are longer than the transmission time (given the port speed) of the preceding frame in the sequence.

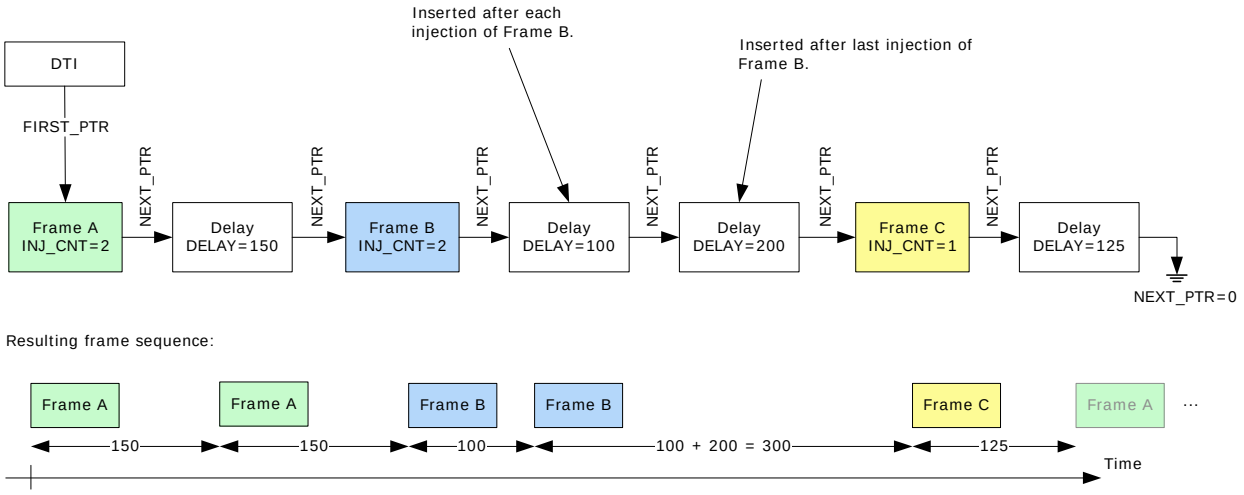
If a DTI sequence is configured with two consecutive frame entries, then these will be injected with a minimum delay of approximately 14 clock cycles.

3.27.3.1 Frame-Delay Sequence

A DTI flow typically consists of a sequence of alternating frame/delay entries in the frame table. If a given frame is to be injected multiple times (FRM_ENTRY_PART0.INJ_CNT>1), then the delay of the following entry in the sequence is applied after each injection. If additional delay is required after the last injection,

then this can be achieved by an additional delay entry. The use of delay entries is shown in the following illustration.

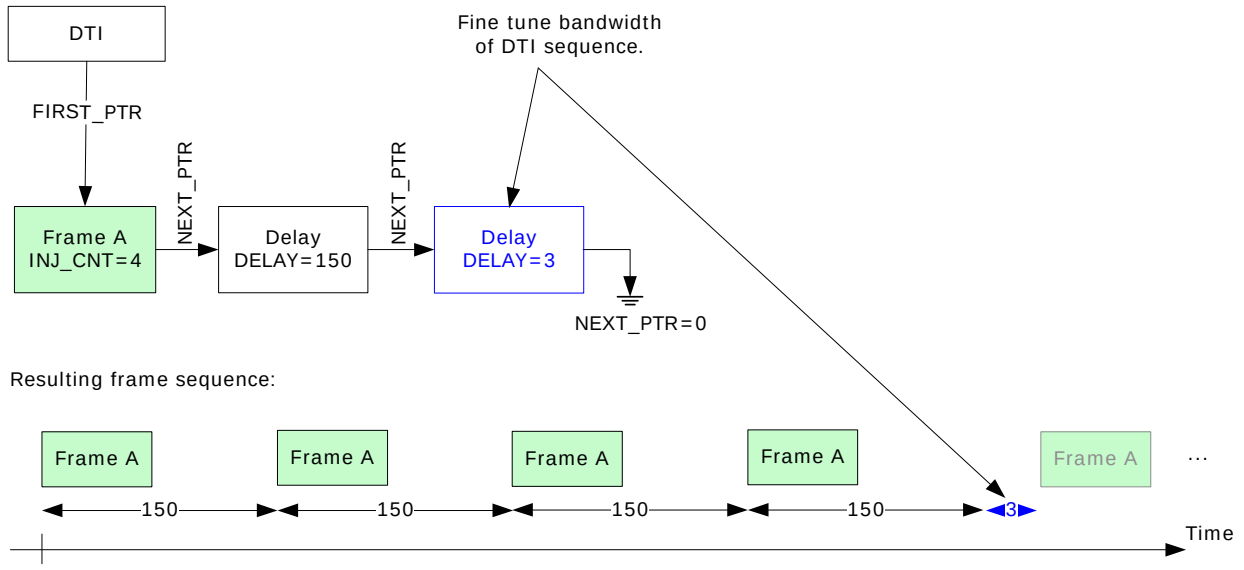
Figure 106 • **DTI Frame/Delay Sequence Example**



3.27.3.2 Bandwidth Fine Tuning

The combination of INJ_CNT and two following delay entries can be used to fine tune the rate generated by a DTI flow. This is shown in the following illustration, where a small delay after the four injections of frame A is used to fine tune the rate of the DTI flow.

Figure 107 • **Fine Tuning Bandwidth of DTI Sequence**



By combining the setting of INJ_CNT and the DELAY value of the last delay depicted, the bandwidth of the DTI sequence can be controlled with high granularity, because the additional (blue) delay will only be applied for every INJ_CNT injections of frame A.

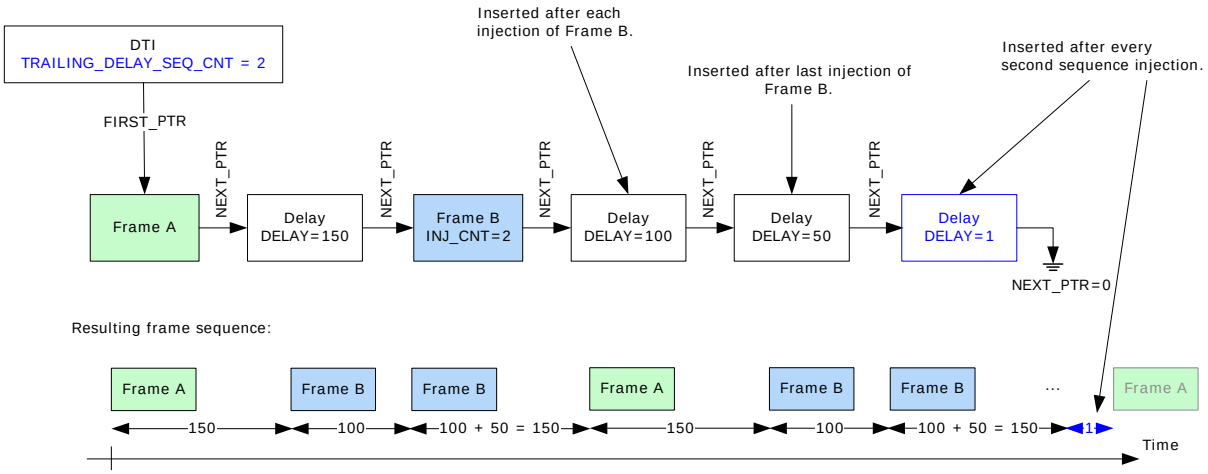
The bandwidth adjustment mechanism shown may be insufficient for controlling the bandwidth of multi-frame sequences (sequences with different frames), because the last delay will be applied for every injection of the frame sequence.

3.27.3.3 Fine Tuning Bandwidth of Multiframe Sequence

For multiframe sequences, additional fine tuning of the bandwidth of the entire sequence can be achieved by configuring the last delay in the sequence, also known as the “trailing delay”, to only be

applied for every Nth injection of the sequence. This is achieved using the TRAILING_DELAY_SEQ_CNT parameter and is shown in the following illustration.

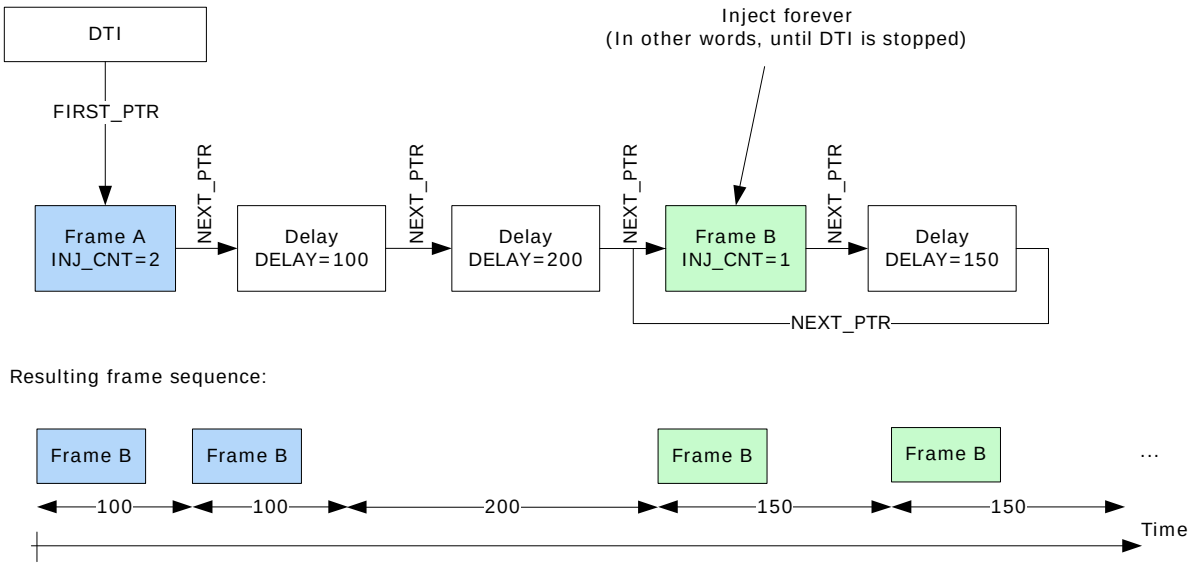
***Figure 108* • Fine Tuning Bandwidth of Multiframe DTI Sequence**



3.27.3.4 Burst Size Test Using Single DTI

If a NEXT_PTR is set to point back to a previous entry in the sequence, then injection will continue forever, that is, until the DTI is stopped. This is depicted in the following illustration. The configuration shown can be used for leaky bucket burst size testing, where the first part of the sequence empties the bucket, followed by a steady frame flow with the rate of the leaky bucket.

***Figure 109* • DTI Frame/Delay Sequence Using Inject Forever**



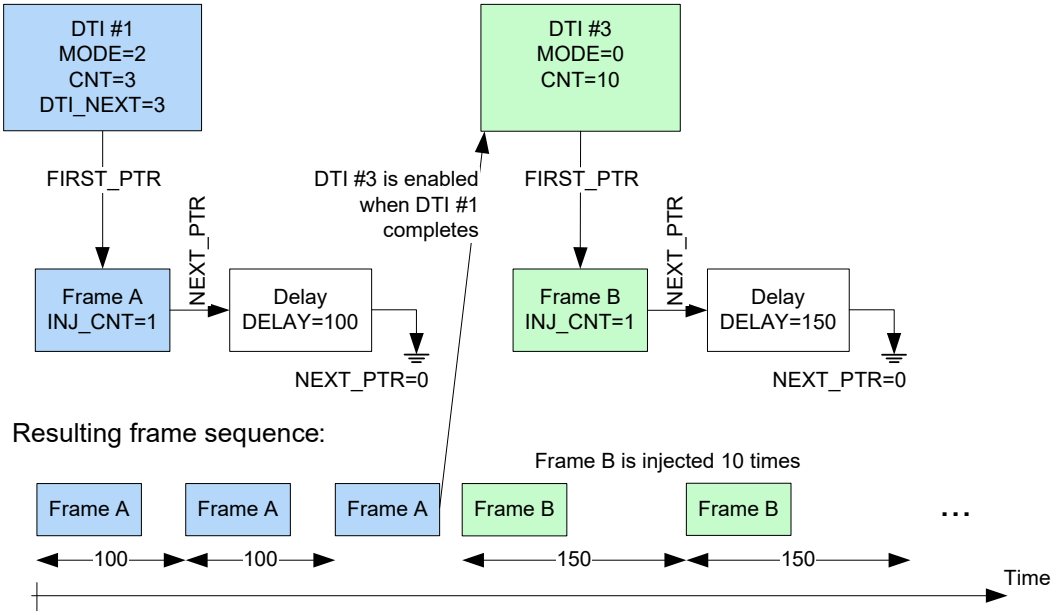
3.27.3.5 Burst Size Test Using DTI Concatenation

Two (or more) DTIs can be concatenated, such that when one DTI completes, another DTI is activated. This is depicted in the following illustration. It can be used for leaky bucket burst size testing, where the first part of the sequence empties the bucket, followed by a steady frame flow with the rate of the leaky bucket.

Note that when DTI concatenation is used, the last delay entry in the frame-delay sequence of the first DTI is not awaited before starting the next DTI. That is, the next DTI is started when the first DTI reads an entry with NEXT_PTR = 0. As shown in the illustration, the delay between the last Frame A and the first Frame B will not be 150 clock cycles, but instead approximately 12 clock cycles. If this is considered

problematic, it can be addressed by using an additional DTI with a sequence consisting only of two delay entries. For example, DTI #2 could be inserted between DTI#1, and DTI#3 and DTI#2 could then be configured with a sequence consisting of two delay entries, with the first having delay = 150.

***Figure 110* • DTI Concatenation**



3.27.3.6 DTI Bandwidth

For 64 bytes frames, AFI can inject up to 10.5 Gbps at 156 MHz clock frequency and up to 3.25 Gbps at 52 MHz clock frequency for a single DTI flow or inject a total bandwidth, across all DTI flows, of up to 26 Gbps at 156 MHz and up to 8 Gbps at 52 MHz.

Higher injection bandwidths can be supported for frame sizes larger than 64 bytes.

Injections into ingress data path must be forwarded through VD1. For 64 bytes frames, the maximum bandwidth through VD1 is 10.5 Gbps at 156 MHz and up to 3.25 Gbps at 52 MHz.

Note: Depending on port configuration, cell bus bandwidth may impose additional bandwidth limitations.

The bandwidth that a DTI is injecting into a queue must not exceed the bandwidth, that HSCH is transmitting from the queue, because the port's FRM_OUT_MAX will then be reached, possibly affecting other DTIs injecting on the same port. For more information, see Port Parameters, page 321. The only exception is if there is only one DTI injecting on the port, because there are no other DTIs that could be affected.

3.27.3.7 DTI Sequence Processing Time

For high bandwidth DTI flows, the time it takes to process the delays and frames in a DTI sequence must be taken into account:

- Processing a frame entry with no succeeding delay entry takes 12 clock cycles.
- Processing a frame entry with a succeeding delay entry takes 12 clock cycles.
- Processing a delay entry, which does not succeed a frame entry takes 12 clock cycles.

For example, a sequence consisting of a frame entry followed by two delay entries will take 24 clock cycles to process. If the frame is a 64-byte frame and the clock period is 4 ns, this will corresponds to a maximum flow bandwidth of approximately 7 Gbps ((64 + 20) × 8 bits/(24 × 4 ns) = 7 Gbps), regardless of the delay values configured for the two delays.

Although it takes 12 clock cycles to process a delay entry, it is valid to configure a delay entry with a delay smaller than 12 clock cycles, because this will be compensated for when generating the delay between later frames in the sequence. For more information, see Figure 108, page 313. For example, the delay between the rightmost frame B and the succeeding frame A will be 150 + 12 clock cycles

(instead of 150 + 1). To compensate for this, however, the delay between the succeeding two frames A will be 150 – 11 clock cycles.

3.27.4 Timer Triggered Injection

For TTIs, frame injection is controlled by timers. To control the time between injections, configure the following registers for each TTI:

- TTI_TIMER.TICK_IDX. For each TTI, one of eight configurable timer ticks shown in the following table must be chosen.
- TTI_TIMER.TIMER_LEN. The number of timer ticks that must elapse between each injection.

That is, the period between each injection becomes tick_period × TIMER_LEN.

The following table lists the registers associated with TTI timer tickers.

Table 159 • **TTI Timer Ticks Registers Overview**

Register	Field	Description	Replication
TTI_TICK_LEN_0_3 TTI_TICK_LEN_4_7	LEN0 - LEN7	Length of timer ticks. The length of each of the eight timer ticks is specified as multiples of the length of the preceding timer tick (that is, <tick_idx>-1). Timer 0 is the shortest timer tick, and timer tick 7 is the longest. Tick 0 is specified in multiple of TTI_TICK_BASE (default 52 µs). Default configuration: **tick_idx** **tick_len** **Tick_Period** 0 1 52 µs (1 × 52 µs) 1 8 416 µs (8 × 52 µs) 2 8 3.3 ms (8 × 416 µs) 3 3 10 ms (3 × 3.3 ms) 4 10 100 ms (10 × 10 ms) 5 10 1 second (10 × 100 ms) 6 10 10 seconds (10 × 1 sec) 7 6 1 minute (6 × 10 sec)	8
TTI_TICK_BASE	BASE_LEN	Length of TTI base tick in system clock cycles.	1

The following table lists the registers associated with configuring parameters for each of the 2K entries in the TTI table.

Table 160 • **TTI Parameters (2048 entries)**

Register	Field	Description	Replication
TTI_PORT_QU	PORT_NUM	Port number on which injection queue transmits.	Per TTI
TTI_TIMER	TICK_IDX	Which of the eight timer ticks is used by the timer controlling the injection of this frame.	Per TTI
TTI_TIMER	JITTER_MODE	Enables jitter for the number of timer ticks between each injection of this frame. 0: No jitter. 1: Timer is set to a random value in the range [TIMER_LEN*0.75; TIMER_LEN]. 2: Timer is set to a random value in the range [TIMER_LEN*0.50; TIMER_LEN]. 3: Timer is set to a random value in the range [1;TIMER_LEN].	Per TTI

Table 160 • **TTI Parameters (2048 entries) *(continued)***

Register	Field	Description	Replication
TTI_TIMER	TIMER_LEN	Number of timer ticks between each injection of this frame. 0: Disables timer. 0xff: Injects frame next time the entry is serviced. After servicing, hardware sets TIMER_LEN to 0. This is intended to be used during removal of frame from buffer memory. See Removing Injection Frames, page 321.	Per TTI
TTI_FRM	FRM_PTR	Points to the frame (in frame table), which the TTI injects.	Per TTI
TTI_TICKS	TICK_CNT	Number of ticks until next injection. Should be set to a random value in the range 1-TIMER_LEN before starting TTI.	Per TTI
TTI_MISC_CFG	INJ_CNT_ENA	Enables counting of injected frames in TTI_MISC:TTI_INJ_CNT.TTI_INJ_CNT.	Per TTI

An entry in the TTI table is checked approximately every two clock cycles. If the TTI's tick (configured through TICK_IDX) has changed, then the TICK_CNT is decremented. If it becomes zero, the frame is injected, and TICK_CNT is set to TIMER_LEN.

3.27.4.1 TTI Calendar

In default configuration, TTI table entries are serviced from 0-2047, then restarted again at entry 0.

When configuring the TTI table, the following must be considered:

- It takes up to 4 clock cycles, worst-case, to service an entry in the TTI table.
- A TTI must be serviced more frequently than the frequency of the TTI's tick (see TTI_TIMER.TICK_IDX).

This means that looping through the entire TTI table may take 2048 × 4 = 8192 clock cycles. With a clock period of 4 ns, this corresponds to approximately 66 µs. In other words, no TTI must use a tick faster than 66 us.

If TTIs faster than 66 µs are required, the TTI calendar can be used to have some entries in the TTI table serviced more often than others, meaning the TTI table can be split into sections with TTIs using ticks of different speeds.

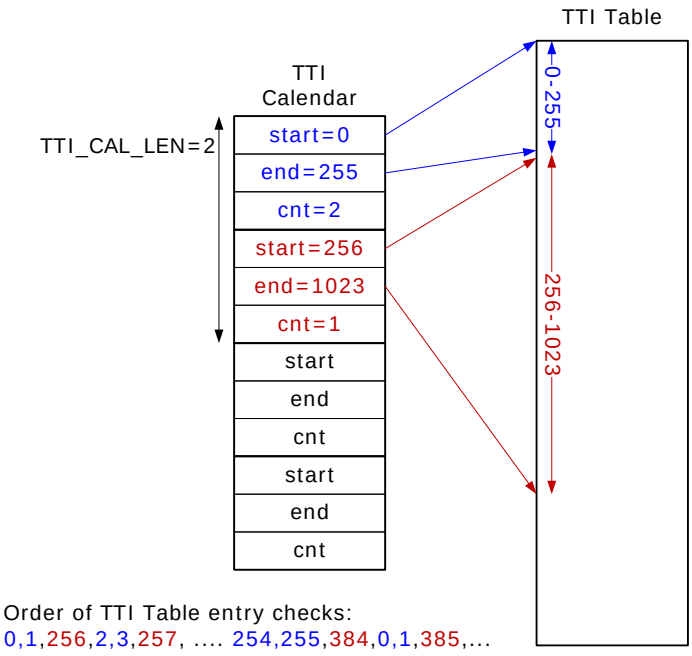
The TTI calendar consists of four entries, as described in Table 6

Table 161 • **TTI Calendar Registers Overview**

Register	Field	Description	Replication
TTI_CAL_SLOT_PTRS	SLOT_START_PTR SLOT_END_PTR	Calendar slot's frame table start and end pointer.	4
TTI_CAL_SLOT_CNT	SLOT_CNT	Number of TTIs to service in slot before moving to next TTI calendar slot.	4
TTI_CTRL	TTI_CAL_LEN	Length of TTI calendar.	1
TTI_CTRL	TTI_INIT	When set, initializes calendar to start at calendar slot 0. Cleared by AFI when done.	1

The following illustration shows a configuration for the TTI calendar. In this case, only TTI entry 0-1023 are being used.

Figure 111 • **TTI Calendar Example**



In the example, it takes (256/2) × (1 + 2) × 4 = 1536 clock cycles to service all TTIs in the blue section. With a clock cycle of 4 ns, the TTIs in the blue section must not use ticks faster than 1536 × 4 = 6144 ns.

Similarly, the 768 TTIs in the red section must not use ticks faster than
(768/1) × (1 + 2) × 4 × 4 ns = ~36.9 µs

3.27.4.2 Other TTI Parameters

Other TTI-related parameters are listed in the following table.

Table 162 • **Miscellaneous TTI Registers Overview**

Register	Field	Description	Replication
TTI_CTRL	TTI_ENA	Enables TTI. Before enabling TTI, TTI_INIT should be used to initialize calendar state.	1
TTI_INJ_CNT	TTI_INJ_CNT	Number of TTI injections. Enabled per TTI using TTI_TBL:TTI_MISC_CFG.INJ_CNT_ENA.	1

3.27.4.3 TTI Table Update Engine (AFI TUPE)

The AFI Table Update Engine (AFI TUPE) can be used to quickly update a large number of entries in the TTI Table such as disabling or enabling timers in fail-over scenarios. AFI TUPE related parameters are listed in the following table. The parameters prefixed with CRIT are used to specify the criteria, which TTIs fulfill in order for TUPE to apply.

Table 163 • **AFI TUPE Registers Overview**

AFI:TUPE Register	Field	Description	Replication
TUPE_MISC	TUPE_START	Start TUPE.	1
TUPE_MISC	CRIT_TUPE_CTRL_VAL_ENA	Enable use of CRIT_TUPE_CTRL_VAL and CRIT_TUPE_CTRL_MASK.	1
TUPE_MISC	CRIT_PORT_NUM_ENA	Enable use of CRIT_PORT_NUM_VAL.	1

Table 163 • **AFI TUPE Registers Overview *(continued)***

AFI:TUPE Register	Field	Description	Replication
TUPE_MISC	CRIT_QU_NUM_ENA	Enable use of CRIT_QU_NUM_VAL.	1
TUPE_MISC	CMD_TIMER_ENA_ENA	Enable use of CMD_TIMER_ENA_VAL.	1
TUPE_MISC	CMD_TIMER_ENA_VAL	TUPE command parameter: New TIMER_ENA value.	1
TUPE_MISC	CMD_PORT_NUM_ENA	Enable use of CMD_PORT_NUM_VAL.	1
TUPE_MISC	CMD_QU_NUM_ENA	Enable use of CMD_QU_NUM_VAL.	1
TUPE_ADDR	TUPE_START_ADDR	First address in TTI table for AFI TUPE to process.	1
TUPE_ADDR	TUPE_END_ADDR	Last address in TTI table for AFI TUPE to process.	1
TUPE_CRIT1	CRIT_PORT_NUM_VAL	TUPE criteria parameter controlling which TTI table entries to update.	1
TUPE_CRIT1	CRIT_QU_NUM_VAL	TUPE criteria parameter controlling which TTI table entries to update.	1
TUPE_CRIT2	CRIT_TUPE_CTRL_MASK	TUPE criteria parameter controlling which TTI table entries to update.	1
TUPE_CRIT3	CRIT_TUPE_CTRL_VAL	TUPE criteria parameter controlling which TTI table entries to update.	2
TUPE_CMD1	CMD_PORT_NUM_VAL	TUPE command parameter: New PORT_NUM value.	1
TUPE_CMD1	CMD_QU_NUM _VAL	TUPE command parameter: New QU_NUM value.	1

The following TTI parameters can be used as part of the TUPE criteria:

- PORT_NUM
- QU_NUM
- TUPE_CTRL

The parameters prefixed "CMD" specify the commands, which are applied to any TTIs, which fulfill the TUPE criteria. Using the TUPE command parameters, the following TTI parameters can be changed:

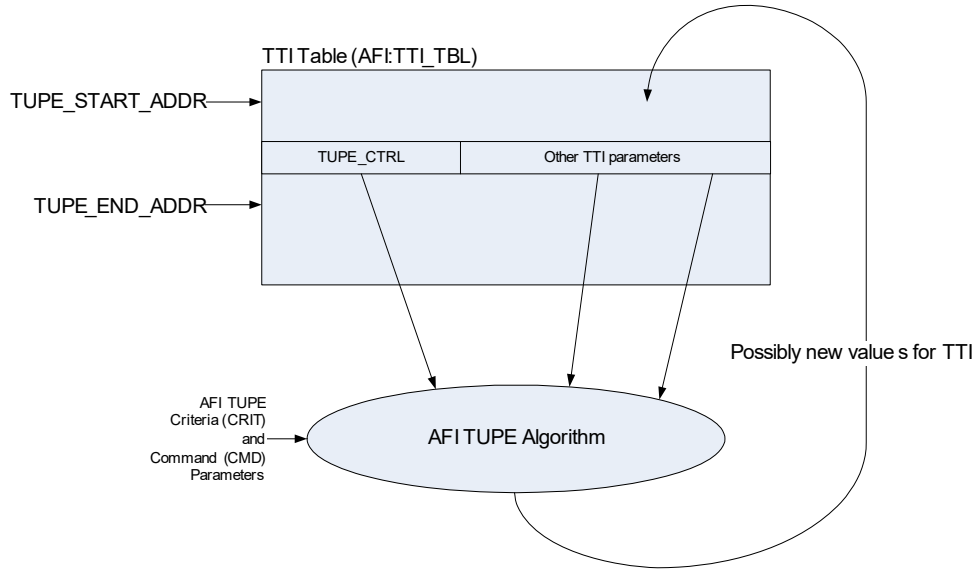
- TIMER_ENA
  Timers can be enabled/disabled.
- PORT_NUM
  Injections can be moved to a different port.
- QU_NUM
  Injections can be moved to a different queue.

While TUPE is running no injections will be performed for any TTIs which match the configured TUPE criteria and fall within the configured TUPE address range.

The TUPE_CTRL field in the TTI Table can be used to classify TTIs into different groups, such that a TUPE command can easily address all TTIs within such group.

The following illustration depicts the AFI TUPE functionality.

Figure 112 • **AFI TUPE**



The full algorithm used by AFI TUPE is shown in the following pseudo code. Configuration parameters are prefixed "csr.".

```
// AFI Table UPdate Engine (AFI TUPE) Algorithm

bool tupe_ctrl_match;

for (addr = csr.tupe_start_addr; addr < csr.tupe_end_addr; addr++) {
  tti_tbl_entry = tti_tbl[addr];

  tupe_ctrl_match = FALSE;
  for (i = 0; i < 2; i++) {
    if ((tti_tbl_entry.tupe_ctrl & csr.crit_tupe_ctrl_mask) ==
        csr.crit_tupe_ctrl_val[i]) {
      tupe_ctrl_match = TRUE;
    }
  }

  if (
      (// Check matching TUPE_CTRL value
       tupe_ctrl_match)
      &&
      (// If enabled, check if TTI's PORT_NUM matches csr.crit_port_num_val
       !csr.crit_port_num_ena
       ||
       (tti_tbl_entry.port_num == csr.crit_port_num_val))
      &&
      (// If enabled, check if TTI's QU_NUM matches csr.crit_qu_num_val
       !csr.crit_qu_num_ena
       ||
       (tti_tbl_entry.qu_num == csr.crit_qu_num_val))
      ) {
    // TTI fulfills criterias => Apply TUPE command to TTI

    // timer_ena
    if (csr.cmd_timer_ena_ena) {
      tti_tbl_entry.timer_ena = csr.cmd_timer_ena_val;
```

```
    }

    // port_num
    if (csr.cmd_port_num_ena) {
      tti_tbl_entry.port_num = csr.cmd_port_num_val;
    }

    // qu_num
    if (csr.cmd_qu_num_ena) {
      tti_tbl_entry.qu_num = csr.cmd_qu_num_val;
    }

    // Write back to TTI table
    tti_tbl[addr] = tti_tbl_entry;
  }
}
```

3.27.5 Injection Queues

DTI and TTI may inject into any egress queue in the queue system. In injection context, the queues can be divided into the following categories:

- Normal queues. These queues are processed by each level of SEs in HSCH. Frames injected into such queues are thus subject to the M-DWRR and shaping algorithms of the SEs. Frames are injected into the tail of the selected queue.
- Port injection queue without shaping. These queues, one for each port, inject frames on the port with higher priority than normal queues. The state of the DLB shaper is disregarded and is not updated with the size of the transmitted frame.

  Injected frames are injected into the tail of the queue, but due to the high priority, the queue will normally be (almost) empty.
- Port injection queue with shaping. These queues, one for each port, inject frames on the port with higher priority than normal queues. The state of the port shaper is respected and is updated with the size of the transmitted frame.

  Injected frames are injected into the tail of the queue, but due to the high priority, the queue will normally be (almost) empty.

The queue into which a TTI or DTI injects frames is controlled by the QU_NUM parameter. The actual value to be used for these parameters depend on the HSCH configuration. For more information, see Queue Mapping, page 296.

For injections into the ingress data path, frames must be looped through VD1 and QU_NUM must be configured to select a VD1 queue.

3.27.6 Adding Injection Frame

To add a frame for injection by AFI, the CPU must follow this procedure:

1. Set AFI::MISC_CTRL.AFI_ENA to 1 before any use of AFI.
2. Send the frame to AFI. AFI_INJ must be set in the IFH. For more information, see Frame Headers, page 22.
3. Poll AFI::NEW_FRM_CTRL.VLD to await the frame being received by AFI.
4. When the frame has been received by AFI, then copy frame information from AFI::NEW_FRM_INFO to an unused entry in the Frame table.
5. Clear AFI::NEW_FRM_CTRL.VLD.
6. TTI: Set up the TTI table to have the frame injected with the desired interval and to the desired queue and port. For more information, see Timer Triggered Injection, page 315.
7. DTI: For injection of a sequence of frames, repeat steps 1 through 4.

   Set up the DTI Table to inject the frames. For more information, see Table 157, page 311.

Configure other DTI parameters. For more information, see Table 158, page 311.

3.27.7 Starting Injection

TTI is enabled by using TTI_INIT to initialize and then setting TTI_ENA = 1.

Each TTI is started by setting TIMER_LEN a non-zero value.

Each DTI is started by setting DTI_CTRL.ENA = 1.

3.27.8 Stopping Injection

Each TTI is stopped by setting TIMER_LEN to 0.

All TTIs can be stopped by setting TTI_ENA = 0.

Each DTI is stopped by setting DTI_CTRL.ENA = 0.

3.27.9 Removing Injection Frames

This section provides information about removing a single frame (TTI) or frame sequence (DTI) from the buffer memory.

3.27.9.1 Single Frame (TTI)

To remove a single frame from buffer memory, the frame must be injected one more time. This “removal injection” does not result in a frame transmission, but purely serves to remove the frame from the buffer memory.

Use the following procedure to remove a single frame from buffer memory.

1. Set the FRM_RM bit for frame in the frame table.
2. For TTI: Set TIMER_LEN to 0x1ff.
3. Poll the FRM_GONE bit until it gets set by AFI.

When performing removal injection, injection must be enabled for the corresponding port by setting AFI:PORT_TBL[<port>]:PORT_CFG.FRM_OUT_MAX != 0.

3.27.9.2 Frame Sequence (DTI)

Use the following procedure to remove a DTI frame sequence from buffer memory.

1. Disable the DTI by setting DTI_CTRL.ENA = 0.
2. Set the FRM_RM bit for each frame to be removed in the frame table.
3. Set DTI_FRM.NEXT_FRM_PTR to DTI_FRM.FIRST_FRM_PTR.
4. Set DTI_MODE.MODE = 0.
5. Set DTI_CNT.CNT = 1.
6. Set DTI_MODE.FRM_INJ_CNT to 0.
7. Optionally, set all delays in sequence to 0 to speed up the removal procedure.
8. Set DTI_CNT_DOWN.CNT_DOWN to 0.
9. Enable the DTI by setting DTI_CTRL.ENA = 1.
10. Poll the FRM_GONE for the last frame to be removed until the bit gets set by AFI.

This procedure causes the frames to be injected one last time and through this, be removed from buffer memory. The frames will not actually be transmitted on the destination port.

When performing removal injection, injection must be enabled for the corresponding port by setting AFI:PORT_TBL[<port>]:PORT_CFG.FRM_OUT_MAX != 0.

3.27.10 Port Parameters

To ensure that the queue system does not overflow with injected frames, for example, if a port is in flow control, an upper bound on the number of injected, but not yet transmitted, frames per port can be configured in FRM_OUT_MAX.

If FRM_OUT_MAX is reached, then any DTI injections are postponed until the number of outstanding frames falls below FRM_OUT_MAX. TTI injections uses a slightly higher limit of

FRM_OUT_MAX+TTI_FRM_OUT_MAX. This ensures that TTI injection are still possible, even if a DTI flow faster than the port speed has been configured.

If PFC is used in conjunction with automatic frame injection, then either all frames must be injected into queues controlled by the same flow control priority, or all frames must be injected into queues that cannot be stopped by PFC.

If EEE is used in conjunction with automatic frame injection, then all frames for that port must be injected into queues with the same urgent configuration.

Setting FRM_OUT_MAX = 0 will stop all injections on port.

Table 164 • **Port Parameters**

Register	Field	Description	Replication
PORT_CFG	FRM_OUT_MAX	Maximum number of injections outstanding for the port at-a-time.	Per port
PORT_CFG	FRM_RM_ONLY	Only allows frame removal injections. That is, normal injections are not allowed.	Per port
TTI_PORT_FRM_OUT	TTI_FRM_OUT_MAX	Additional injections that can be outstanding before affecting TTI injection.	Per port

3.28 Rewriter

The switch core includes a rewriter (REW) that is common for all ports. The rewriter performs all frame editing prior to frame transmission. Each frame can be handled multiple times by the rewriter

- One time per destination port.
- One time towards the mirror target port.
- One time when copied or redirected to internal and/or external CPU.
- One time when looped back to the analyzer.

All frames that are input to the rewriter can be modified by previous blocks of the frame processing flow. The internal frame header informs the rewriter what ingress frame manipulations (popping) are performed on any given frame. The same ingress frame manipulation is performed for each copy of a frame, while the rewriter may perform per destination specific egress frame manipulations (pushing) to the frame.

The rewriter is the final step in the frame processing flow of the device.

3.28.1 Rewriter Operation

The rewriter supports the following frame operations.

- VLAN tag (802.1Q, 802.1ad) editing: tagging of frames and remapping of PCP and DEI.
- L3 routing. SMAC/DMAC, VID and TTL modifications for IPv4 and IPv6.
- Y1731 and MPLS-TP OAM editing and statistics by interfacing to the VOP.
- Implement Precision Time Protocol (PTP) time stamping by interfacing to the PTP block.
- Interfacing to the loopback block. Loopback frames to ANA based on VOP or VCAP_ES0 actions. Rewrite MACs and IFH of looped frames.
- DSCP remarking: rewriting the DSCP value in IPv4 and IPv6 frames based on classified DSCP value.
- MPLS editing: encapsulation of MPLS link layer and MPLS label stack.
- Insert or update VSTAX headers.
- Handling of both Rx and Tx mirror frames.
- Padding of undersize frames to 64 bytes or 76 bytes.
- Frame FCS update control.
- Add preamble to all frames with an external port destination.
- Update Departure Service Point Statistics.

3.28.2 Supported Ports

The device operates with up to 11 physical ports, two loopback ports, and two DEVCPU extraction ports as possible frame destinations. These frame destinations are addressed by the rewriter using the egress port number.

3.28.3 Supported Frame Formats

The rewriter can identify and rewrite both headers and payload of the following frame types.

Ethernet Frames

- Ethernet link layer DMAC+SMAC
- Maximum of three VLAN Q-tags (802.1Q, 802.1ad)
- Payload (IPv4, IPv6, Y.1731 OAM, PTP, and so on)

Ethernet over MPLS over Ethernet

- MPLS link layer DMAC+SMAC
- Maximum of three VLAN Q-tags in MPLS link layer header
- Maximum of three MPLS labels
- One optional control word (CW) for pseudo wire (PW)
- Ethernet link layer DMAC+SMAC
- Maximum three VLAN Q-tags in Ethernet link layer
- Payload (IPv4, IPv6, Y.1731 OAM, PTP, and so on)

MPLS over Ethernet

- MPLS link layer DMAC+SMAC
- Maximum of three VLAN Q-tags in MPLS link layer headers
- Maximum of three MPLS labels
- One control word
- Payload (MPLS-TP OAM, IPv4 PTP, IPv6 PTP)

The analyzer places information in the IFH to help the rewriter with frame identification. The IFH fields, IFH.DST.ENCAP.W16_POP_CNT and IFH.DST.ENCAP.TYPE_AFTER_POP, are used for this purpose.

3.28.3.1 Maximum Frame Expansion

The rewriter can push 60 bytes into a frame when the IFH is popped and when no other frame pop operations are done. The frame preamble will occupy 8 of the 60 bytes that can be pushed.

3.28.4 Rewriter Initialization

Rewriter initialization comprises two steps:

- Set REW::RAM_INIT.RAM_ENA to 1
- Set REW::RAM_INIT.RAM_INIT to 1

Before the rewriter can be configured, the RAM-based configuration registers must be initialized by writing a 1 to the above RAM_INIT registers. The RAM_INIT.RAM_INIT register is reset by hardware when the initialization is complete.

If VCAP_ES0 is enabled, it must be initialized. All of the default port actions in VCAP_ES0 should also be initialized to ensure proper operation. For more information about initializing VCAP_ES0, see Versatile Content-Aware Processor (VCAP), page 54.

3.28.5 VCAP_ES0 Lookup

The rewriter performs stage 0 VCAP egress matching (VCAP_ES0) lookups for each frame when enabled. The following table lists the registers associated with VCAP_ES0 lookup.

Table 165 • **ES0 Configuration Registers**

Register	Description	Replication
ES0_CTRL. ES0_LU_ENA	Enables lookup in VCAP_ES0 to control advanced frame modifications.	None

Table 165 • **ES0 Configuration Registers *(continued)***

Register	Description	Replication
ES0_CTRL. ES0_BY_RLEG	Enables ES0 router mode lookup when IFH.FWD.DST_MODE indicates routing.	None
ES0_CTRL. ES0_BY_RT_FWD	Enables ES0 router mode lookup when IFH.DST.ENCAP.RT_FWD indicates routing.	None
PORT_CTRL. ES0_LPORT_NUM	Maps the configuration port to a logical port number to be used by ES0 keys. The port used in the lookup can be Tx-mirrored.	Ports

VCAP_ES0 lookup is enabled by setting REW::ES0_CTRL.ES0_LU_ENA = 1. Frames destined to the DEVCPU cannot be rewritten, and there is no VCAP_ES0 lookup for these frames.

All ES0 actions are ignored when VCAP_ES0 is disabled.

There are two ES0 key types: VID and ISDX.

The VID key is always used if:

- IFH.FWD.ES0_ISDX_KEY_ENA = 0
- REW::ES0_CTRL.ES0_BY_RLEG = 1 and (IFH.FWD.DST_MODE = L3UC_ROUTING or IFH.FWD.DST_MODE = L3MC_ROUTING)
- REW::ES0_CTRL.ES0_BY_RT_FWD = 1 and IFH.DST.ENCAP.RT_FWD = 1.

The ISDX key is used if a VID key is not used and IFH.FWD.ES0_ISDX_KEY_ENA = 1.

Table 166 • **VCAP_ES0 Keys**

Key Field	Description	Size	ISDX	VID
X1_TYPE	X1 type. 0: ISDX. 1: VID.	1	x	x
EGR_PORT	Logical egress port number. Configuration port mapped via REW::PORT_CTRL.ES0_LPORT_NUM. The default is no mapping (EGR_PORT = configuration port number). The configuration port is either the physical port or a Tx-mirror port number.	4	x	x
PROT_ACTIVE	Protection is active. Value is IFH.ENCAP.PROT_ACTIVE.	1	x	x
VSI	Virtual Switch Instance. Default value is VSI = IFH.DST.ENCAP.GEN_IDX if IFH.DST.ENCAP.GEN_IDX_MODE = 1 else 0. The following conditions will change the default: If REW::ES0_CTRL.ES0_BY_RLEG = 1 and (IFH.FWD.DST_MODE = L3UC_ROUTING or IFH.FWD.DST_MODE = L3MC_ROUTING) Force VSI = 0x3ff to identify routing. If REW::ES0_CTRL.ES0_BY_RT_FWD = 1 and IFH.ENCAP.RT_FWD = 1. Force VSI = 0x3ff to identify routing.	7	x	x
COSID	Class of Service Identifier. Value is IFH.VSTAX.MISC.COSID if REW::PORT_CTRL.VSTAX2_MISC_ISDX_ENA = 1 else 0.	3	x	X
COLOR	Frame color. Value is IFH.VSTAX.QOS.DP mapped through REW::DP_MAP.DP.	1	x	x
SERVICE_FRM	Service frame. Set to 1 if IFH.VSTAX.MISC.ISDX > 0 and REW::PORT_CTRL.VSTAX2_MISC_ISDX_ENA = 1 else 0.	1	x	x

Table 166 • **VCAP_ES0 Keys *(continued)***

Key Field	Description	Size	ISDX	VID
ISDX	Ingress Service Index. Value is IFH.VSTAX.MISC.ISDX if REW::PORT_CTRL.VSTAX2_MISC_ISDX_ENA = 1 else 0.	9	x	
VID	IEEE VLAN identifier. Default value is VID = IFH.VSTAX.TAG.VID. The following conditions will change the default: If REW::ES0_CTRL.ES0_BY_RLEG = 1 and (IFH.FWD.DST_MODE = L3UC_ROUTING or IFH.FWD.DST_MODE = L3MC_ROUTING) VID = IFH.DST.L3[UC|MC].ERLEG If REW::ES0_CTRL.ES0_BY_RT_FWD = 1 and IFH.DST.ENCAP.RT_FWD = 1 VID = IFH.DST.ENCAP.GEN_IDX.	12		x

The following tables show the actions available in VCAP_ES0. Default actions are selected by PORT_CTRL.ES0_LPORT_NUM when no ES0 entry is hit.

When REW::ES0_CTRL.ES0_LU_ENA = 1, all of the default actions must be initialized to ensure correct frame handling.

Table 167 • **VCAP_ES0 Actions**

Field Name	Description	Width
PUSH_OUTER_TAG	Controls tag A (outer tagging). 0: Port tagging. Push port tag as outer tag if enabled for port. No ES0 tag A. 1: ES0 tag A. Push ES0 tag A as outer tag. No port tag. 2: Forced port tagging. Push port tag as outer tag. No ES0 tag A. 3: Forced untagging. No outer tag pushed (no port tag, no ES0 tag A).	2
PUSH_INNER_TAG	Controls tag B (inner tagging). 0: Do not push ES0 tag B as inner tag. 1: Push ES0 tag B as inner tag.	1

TAG_A_TPID_SEL	Selects TPID for ES0 tag A.
	0: 0x8100.
	1: 0x88A8.
	2: Custom1.
	3: Custom2.
	4: Custom3.
	5: Classified. Selected by ANA in IFH.VSTAX.TAG_TYPE and IFH.DST.ENCAP.TAG_TPID:
	If IFH.DST.ENCAP.TAG_TPID = STD_TPID:
	  If IFH.VSTAX.TAG.TAG_TYPE = 0, then 0x8100 else 0x88A8
	If IFH.DST.ENCAP.TAG_TPID = CUSTOM[*n*]:
	  REW::TPID_CFG[*n*]:TPID_VAL.
		3

TAG_A_VID_SEL	Selects VID for ES0 tag A. 0: Classified VID + VID_A_VAL. 1: VID_A_VAL.	1

Table 167 • **VCAP_ES0 Actions *(continued)***

Field Name	Description	Width
TAG_A_PCP_SEL	Select PCP source for ES0 tag A. 0: Classified PCP. 1: PCP_A_VAL. 2: Reserved. 3: PCP of popped VLAN tag if available (IFH.VSTAX.TAG.WAS_TAGGED = 1 and num_popped_tags>0) else PCP_A_VAL. 4: Mapped using mapping table 0, otherwise use PCP_A_VAL. 5: Mapped using mapping table 1, otherwise use mapping table 0. 6: Mapped using mapping table 2, otherwise use PCP_A_VAL. 7: Mapped using mapping table 3, otherwise use mapping table 2.	3
TAG_A_DEI_SEL	Select DEI source for ES0 tag A. 0: Classified DEI. 1: DEI_A_VAL. 2: REW::DP_MAP.DP [IFH.VSTAX.QOS.DP]. 3: DEI of popped VLAN tag if available (IFH.VSTAX.TAG.WAS_TAGGED = 1 and num_popped_tags>0) else DEI_A_VAL. 4: Mapped using mapping table 0, otherwise use DEI_A_VAL. 5: Mapped using mapping table 1, otherwise use mapping table 0. 6: Mapped using mapping table 2, otherwise use DEI_A_VAL. 7: Mapped using mapping table 3, otherwise use mapping table 2.	3
TAG_B_TPID_SEL	Selects TPID for ES0 tag B. 0: 0x8100. 1:0x88A8. 2: Custom 1. 3: Custom 2. 4: Custom 3. 5: See TAG_A_TPID_SEL.	3
TAG_B_VID_SEL	Selects VID for ES0 tag B. 0: Classified VID + VID_B_VAL. 1: VID_B_VAL.	1
TAG_B_PCP_SEL	Selects PCP source for ES0 tag B. 0: Classified PCP. 1: PCP_B_VAL. 2: Reserved. 3: PCP of popped VLAN tag if available (IFH.VSTAX.TAG.WAS_TAGGED=1 and num_popped_tags>0) else PCP_B_VAL. 4: Mapped using mapping table 0, otherwise use PCP_B_VAL. 5: Mapped using mapping table 1, otherwise use mapping table 0. 6: Mapped using mapping table 2, otherwise use PCP_B_VAL. 7: Mapped using mapping table 3, otherwise use mapping table 2.	3
TAG_B_DEI_SEL	Selects DEI source for ES0 tag B. 0: Classified DEI. 1: DEI_B_VAL. 2: REW::DP_MAP.DP [IFH.VSTAX.QOS.DP]. 3: DEI of popped VLAN tag if available (IFH.VSTAX.TAG.WAS_TAGGED = 1 and num_popped_tags>0) else DEI_B_VAL. 4: Mapped using mapping table 0, otherwise use DEI_B_VAL. 5: Mapped using mapping table 1, otherwise use mapping table 0. 6: Mapped using mapping table 2, otherwise use DEI_B_VAL. 7: Mapped using mapping table 3, otherwise use mapping table 2.	3

Table 167 • **VCAP_ES0 Actions *(continued)***

Field Name	Description	Width
TAG_C_TPID_SEL	Selects TPID for ES0 tag C. 0: 0x8100. 1:0x88A8. 2: Custom 1. 3: Custom 2. 4: Custom 3. 5: See TAG_A_TPID_SEL.	3
TAG_C_PCP_SEL	Selects PCP source for ES0 tag C. 0: Classified PCP. 1: PCP_C_VAL. 2: Reserved. 3: PCP of popped VLAN tag if available (IFH.VSTAX.TAG.WAS_TAGGED=1 and num_popped_tags>0) else PCP_C_VAL. 4: Mapped using mapping table 0, otherwise use PCP_C_VAL. 5: Mapped using mapping table 1, otherwise use mapping table 0. 6: Mapped using mapping table 2, otherwise use PCP_C_VAL. 7: Mapped using mapping table 3, otherwise use mapping table 2.	3
TAG_C_DEI_SEL	Selects DEI source for ES0 tag C. 0: Classified DEI. 1: DEI_C_VAL. 2: REW::DP_MAP.DP [IFH.VSTAX.QOS.DP]. 3: DEI of popped VLAN tag if available (IFH.VSTAX.TAG.WAS_TAGGED = 1 and num_popped_tags>0) else DEI_C_VAL. 4: Mapped using mapping table 0, otherwise use DEI_C_VAL. 5: Mapped using mapping table 1, otherwise use mapping table 0. 6: Mapped using mapping table 2, otherwise use DEI_C_VAL. 7: Mapped using mapping table 3, otherwise use mapping table 2.	3
VID_A_VAL	VID used in ES0 tag A. See TAG_A_VID_SEL.	12
PCP_A_VAL	PCP used in ES0 tag A. See TAG_A_PCP_SEL.	3
DEI_A_VAL	DEI used in ES0 tag A. See TAG_A_DEI_SEL.	1
VID_B_VAL	VID used in ES0 tag B. See TAG_B_VID_SEL.	12
PCP_B_VAL	PCP used in ES0 tag B. See TAG_B_PCP_SEL.	3
DEI_B_VAL	DEI used in ES0 tag B. See TAG_B_DEI_SEL.	1
VID_C_VAL	VID used in ES0 tag C. See TAG_C_VID_SEL.	12
PCP_C_VAL	PCP used in ES0 tag C. See TAG_C_PCP_SEL.	3
DEI_C_VAL	DEI used in ES0 tag C. See TAG_C_DEI_SEL.	1
POP_VAL	Pops one additional Q-tag. num_popped_tags = IFH.tagging.pop_cnt + POP_VAL. The rewriter counts the number of Q-tags in the frame and only pops tags that are present in the frame. NUM_POPPED_TAGS is equal to the last available tag.	1
UNTAG_VID_ENA	Controls insertion of tag C. Un-tag or insert mode can be selected. See PUSH_CUSTOMER_TAG.	1
PUSH_CUSTOMER_TAG	Selects tag C mode: 0: Do not push tag C. 1: Push tag C if IFH.VSTAX.TAG.WAS_TAGGED = 1. 2: Push tag C if IFH.VSTAX.TAG.WAS_TAGGED = 0. 3: Push tag C if UNTAG_VID_ENA = 0 or (C-TAG.VID ! = VID_C_VAL).	2

Table 167 • **VCAP_ES0 Actions *(continued)***

Field Name	Description	Width
TAG_C_VID_SEL	Selects VID for ES0 tag C. The resulting VID is termed C-TAG.VID. 0: Classified VID. 1: VID_C_VAL.	1
DSCP_SEL	Selects source for DSCP. 0: Controlled by port configuration and IFH. 1: Classified DSCP via IFH. 2: DSCP_VAL. 3: Reserved. 4: Mapped using mapping table 0, otherwise use DSCP_VAL. 5: Mapped using mapping table 1, otherwise use mapping table 0. 6: Mapped using mapping table 2, otherwise use DSCP_VAL. 7: Mapped using mapping table 3, otherwise use mapping table 2.	3
DSCP_VAL	DSCP value.	6
PROTECTION_SEL	Change ENCAP_ID, ESDX_BASE and OAM_MEP_IDX based on IFH.DST.ENCAP.PROT_ACTIVE. 0: No protection. Do not use protection index values. 1: Use <index>_P if ifh.encap.prot_active = 1. 2: Use <index>_P if ifh.encap.prot_active = 0. 3: Reserved.	2
LABEL_SEL	Selects value of the MPLS pseudo wire label. 0: ES0:MPLS_LABEL. 1-3: Reserved. 4: Mapped using mapping table 0. 5: Mapped using mapping table 1. 6: Mapped using mapping table 2. 7: Mapped using mapping table 3.	3
TC_SEL	Selects TC for MPLS label. 0: Classified TC. 1: TC_VAL. 2: COSID (IFH.VSTAX.MISC.COSID). 3: Reserved. 4: Mapped using mapping table 0, otherwise use TC_VAL. 5: Mapped using mapping table 1, otherwise use mapping table 0. 6: Mapped using mapping table 2, otherwise use TC_VAL. 7: Mapped using mapping table 3, otherwise use mapping table 2.	3
SBIT_SEL	Configures SBIT-selection. 0: Classified SBIT. 1: Fixed. SBIT_VAL.	1
TTL_SEL	Configures TTL selection. 0: Classified TTL. 1: Fixed. TTL_VAL.	1
MPLS_LABEL	MPLS pseudo wire label.	20
TC_VAL	Configured TC value.	3
SBIT_VAL	Configured SBIT value	1
TTL_VAL	Configured TTL value.	8
CW_DIS	Disable control word (CW) in MPLS encapsulation. 0: MPLS_LABEL_CFG.CW_ENA controls CW. 1: CW is disabled.	1

Table 167 • **VCAP_ES0 Actions *(continued)***

Field Name	Description	Width
ENCAP_ID	Index into the MPLS encapsulation table (REW::ENCAP). Setting ENCAP_ID to zero disables pushing of MPLS encapsulation. 0: Disables the use of MPLS encapsulation. 1-127: Selects ENCAP ID.	7
ENCAP_ID_P	Index to use when enabled by PROTECTION_SEL. The encoding is equal to ENCAP_ID.	7
POP_CNT	If POP_CNT > 0, then ifh.dst.encap.w16_pop_cnt is overruled with POP_CNT when popping MPLS labels. The encoding is: 0: Use ifh.dst.encap.w16_pop_cnt as word16 (2 bytes) pop count 1: Do not pop any bytes. 2: Pop 2 × 2 = 4 bytes. … 21: Pop 2 × 21 = 42 bytes. 22-31: Reserved. If the number of bytes popped by POP_CNT is larger than the value given by ifh.dst.encap.w16_pop_cnt, the IFH value is always used.	5
ESDX_BASE	Egress service index. Used to index egress service counter set (REW::CNT_CTRL.STAT_MODE).	10
ESDX_BASE _P	Egress service index to use when enabled by PROTECTION_SEL.	10
ESDX_COSID_ OFFSET	Egress counter set, offset per COSID. Stat index = ESDX_BASE + ESDX_COSID_OFFSET[(3 x COSID + 2) : 3 x COSID]	24
MAP_0_IDX	Index 0 into mapping resource A. Index bits 10:3. Index bits 2:0 are controlled by MAP_0_KEY.	8
MAP_1_IDX	Index 1 into mapping resource A. Index bits 10:3. Index bits 2:0 are controlled by MAP_1_KEY.	8
MAP_2_IDX	Index 0 into mapping resource B. Index bits 10:3. Index bits 2:0 are controlled by MAP_2_KEY.	8
MAP_3_IDX	Index 1 into mapping resource B. Index bits 10:3. Index bits 2:0 are controlled by MAP_2_KEY.	8

Table 167 • **VCAP_ES0 Actions *(continued)***

Field Name	Description	Width
MAP_0_KEY	Key used when looking up mapping table 0. **QoS mappings:** 0: QoS (8 entries): IDX = IDX_A + QOS. 1: DP, QoS (32 entries): IDX = IDX_A + IFH..DP × 8 + QOS. **DSCP mappings:** 2: DSCP (64 entries): IDX = IDX_A + IFH..DSCP. 3: DP, DSCP (256 entries): IDX = IDX_A + 64 × IFH..DP + IFH..DSCP. 4: DP, TOS_PRE (32 entries): IDX = IDX_A + IFH..DP × 8 + TOS_PRE. **PCP mappings:** 5: PCP (8 entries): IDX = IDX_A + IFH..PCP. 6: DP, PCP (32 entries): IDX = IDX_A + IFH..DP × 8 + IFH..PCP. 7: DEI, PCP (16 entries): IDX = IDX_A + IFH..DEI × 8 + IFH..PCP. **TC mappings:** 8: TC: IDX = IDX_A + IFH..TC. 9: DP, TC: IDX = IDX_A + IFH..DP × 8 + IFH..TC. **Popped customer tag mappings:** 10: Popped PCP (8 entries): IDX = MAP_0_IDX + pop.PCP 11: DP, popped PCP (32 entries): IDX = MAP_0_IDX + IFH..DP × 8 + pop.PCP. 12: Popped DEI, popped PCP (16 entries): IDX = MAP_0_IDX + pop.DEI × 8 + pop.PCP. **COSID mappings:** 13: COSID (8 entries): IDX = IDX_A + IFH.VSTAX.MISC.COSID. 14: DP, COSID (32 entries): IDX = IDX_A + IFH..DP × 8 + IFH.VSTAX.MISC.COSID.	4
MAP_1_KEY	See MAP_0_KEY.	4
MAP_2_KEY	See MAP_0_KEY.	4
MAP_3_KEY	See MAP_0_KEY.	4
OAM_MEP_IDX_VLD	Setting this bit to 1 causes the OAM_MEP_IDX to be used.	1
OAM_MEP_IDX	OAM MEP index.	8
OAM_MEP_IDX_P	OAM MEP index to use when enabled by PROTECTION_SEL.	8
OAM_LM_COSID_COLOR_SEL	0: Mapped using mapping table 0 if valid, otherwise use value 0. 1: Mapped using mapping table 1 if valid, otherwise use mapping table 0. 2: Mapped using mapping table 2 if valid, otherwise use value 0. 3: Mapped using mapping table 3 if valid, otherwise use mapping table 2.	2
INDEPENDENT_MEL_ENA	OAM MEL mode.	1
MIP_IDX	Ethernet MIP index. 0: Disable. Other: MIP index value.	7
FWD_SEL	ES0 Forward selector. 0: No action. 1: Copy to loopback interface. 2: Redirect to loopback interface. 3: Discard.	2
CPU_QU	CPU extraction queue. Used when FWD_SEL >0 and PIPELINE_ACT = XTR.	3

Table 167 • **VCAP_ES0 Actions *(continued)***

Field Name	Description	Width
PIPELINE_PT	ES0 pipeline point. 0: REW_IN_SW. 1: REW_OU_SW. 2: REW_SAT. 3: Reserved.	2
PIPELINE_ACT	Pipeline action when FWD_SEL >0. 0: XTR. CPU_QU selects CPU extraction queue 1: LBK_ASM.	1
SWAP_MAC_ENA	This setting is only active when FWD_SEL = 1 or FWD_SEL = 2 and PIPELINE_ACT = LBK_ASM. 0: No action. 1: Swap MACs and clear bit 40 in new SMAC.	1
LOOP_ENA	Setting LOOP_ENA overrides the other pipeline control settings. 0: Forward based on PIPELINE_PT and FWD_SEL 1: Force FWD_SEL=REDIR, PIPELINE_ACT = LBK_ASM, PIPELINE_PT = REW_SAT, swap MACs and clear bit 40 in new SMAC. Do not apply CPU_QU.	1

3.28.5.1 Rewriter Pipeline Handling

In order to achieve the correct statistics and processing of frames, it is important that frames can be extracted, injected, discarded, and looped at specific positions in the processing flow. Based on the position, processing—especially counters, are affected. For information about how pipeline points are used in the processing flow, see Pipeline Points, page 66.

Note that a frame is always physically passed through the entire processing. Logically, certain elements of the processing can be disabled or enabled.

The following table lists the pipeline points available in the rewriter.

Table 168 • **Rewriter Pipeline Point Definitions**

Pipeline Point	What can set it?	Position in Flow
NONE	Default	0
REW_IN_MIP	Inner MIP configured in rewriter	1
REW_IN_SW	Inner software MEP controlled by VCAP_ES0	2
REW_IN_VOE	Service VOE in VOP	3
REW_OU_VOE	Service VOE in VOP	4
REW_OU_SW	Outer software MEP controlled by VCAP_ES0	5
REW_OU_MIP	Outer software MIP configured in rewriter	6
REW_SAT	SAT software MEP controlled by VCAP_ES0	7
REW_PORT_VOE	Port VOE in VOP	8
REW_VRAP	Set in VRAP reply frame	9

Each point in the flow assigns one of the possible pipeline actions listed in the following table.

Table 169 • **Pipeline Actions**

Action Type	Name	Comment
Inject	INJ INJ_MASQ INJ_CENTRAL	Injection actions set by ANA are cleared when the frame enters the rewriter by setting the pipeline point and action to NONE.
Extract	XTR XTR_UPMEP XTR_REW_LATE	XTR is set by ES0 and MIP. XTR_UPMEP is also set by the VOP. XTR_REW_LATE can only be set in ANA.
Loopback	LBK_ASM LBK_QS	LBK_QS is set in ANA. This action is treated as an injection action in the rewriter. LBK_ASM can be set by ES0 when looping frames back to ANA. This action is treated as an extraction in the rewriter.
None	NONE	Default. No pipeline point and action is assigned to frame.

Frames looped by the analyzer (PIPELINE_ACT = LBK_QS) have their ANA pipeline point changed to a REW pipeline point, which indicates the point in the rewriter flow in REW where they appear again after being looped. The following pipeline point translations are handled.

- PIPELINE_PT = ANA_PORT_VOE is changed to REW_PORT_VOE
- PIPELINE_PT = ANA_OU_VOE is changed to REW_OU_VOE
- PIPELINE_PT = ANA_IN_VOE is changed to REW_IN_VOE

Frame forward and counter updates are done based on the assigned pipeline points and actions. In the rewriter, pipeline points and actions can be assigned by ES0, the MIP, and the VOP.

Frames injected by a CPU can also have pipeline point and actions set in the injection IFH.

The rules listed in the following table apply when pipeline points are updated.

Table 170 • **Pipeline Point Updating Rules**

Pipeline Action	Rule
None	The pipeline point and actions can be updated.
Inject	If a frame is injected in pipeline point N, the forwarding cannot be changed in pipeline point N-1 and it will not be counted in the N-1 point. Logically the frame does not exist in the N-1 point. Example: A frame is injected in the REW_VRAP pipeline point (Pipeline point = REW_VRAP, Action = INJ). The SW-MEP is configured to loop the same frame in the pipeline point REW_SAT. The frame will not be looped, because it was injected later in the pipeline (REW_VRAP > REW_SAT). The frame will not be counted by the egress service statistics, which has a default location in the REW_PORT_VOE pipeline point for injected frames.
Extract	If frame is extracted in pipeline point N the forwarding cannot be changed in pipeline point N+1 and it will not be counted in the N+1 point. Logically the frame does not exist in the N+1 point. Example: A frame is extracted in the REW_IN_MIP pipeline point (Pipeline point = REW_IN_MIP, Action = XTR). The SW-MEP is configured to loop the same frame in the pipeline point REW_SAT. The frame will not be looped because it has already been extracted earlier in the pipeline (REW_IN_MIP < REW_SAT). The frame will not be counted by the egress service statistics, which has a default location in the REW_OU_SW pipeline point for extracted frames.

3.28.5.2 Frame Copy, Redirect, or Discard

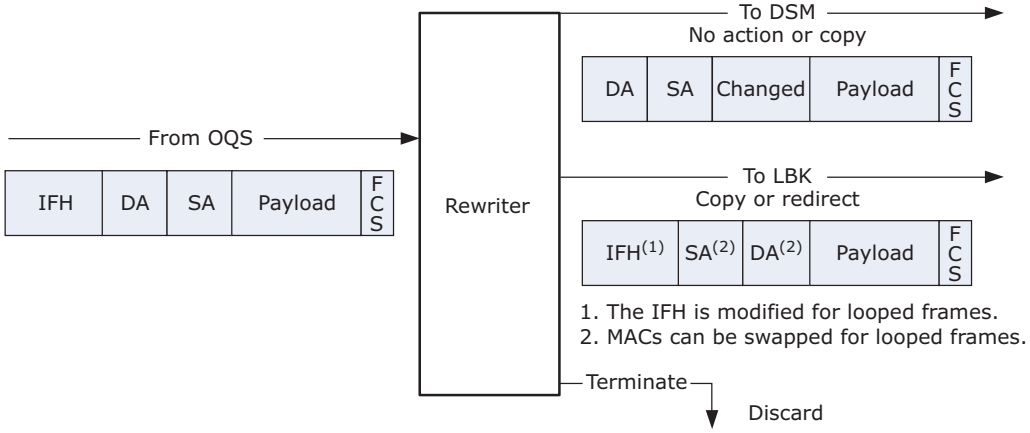
Frame forwarding may be changed by ES0 action FWD_SEL, the MIP or the VOP. Changing frame forwarding will update the frame pipeline point and action.

The following options are available to control frame forwarding.

- No action. Do not change frame forwarding.
- Copy frame to CPU via the loopback interface. The frame is looped back to the ANA where it is sent to the CPU. The IFH of the CPU copy is modified to select a CPU queue. The frame is also forwarded as normal to the destination port and can be modified by the normal rewriter operations.
- Redirect frame to ANA via loopback interface. When doing frame loopbacks, the destination port of the redirected frame will be left unchanged. The IFH of the frame is modified and the MAC addresses may be swapped. Redirected frames may also be sent to a CPU queue. This can be done as a standalone operation or together with the loopback frame. When redirecting, the frame is only sent to the loopback interface and not to the original destination port.
- Discard frame. The frame is terminated before transmission.

The following illustrations depicts options for frame forwarding.

Figure 113 • **Frame Forward Options**



The following table lists the registers associated with the frame loopback options.

Table 171 • **Frame Loopback Options**

Register	Description	Replication
ES0_CTRL. ES0_FRM_LBK_CFG	Using ES0 it is possible to loop frames back to ANA with the LOOP_ENA action. If this bit is set, a frame can only be looped once by ES0.	None

Frame forwarding is changed if allowed by the current frame pipeline point and action. For more information, see Rewriter Pipeline Handling, page 331.

The following table shows the pipeline action when the frame forwarding is changed.

Table 172 • **Pipeline Action Updates**

Forwarding Selection	Pipeline Action Updates
NONE	Unchanged
COPY	Unchanged
REDIR	CPU redirection: XTR / XTR_UPMEP Loopback frames: LBK_ASM
DISCARD	XTR / XTR_UPMEP

The pipeline updating rules ensures that a REDIR or DISCARD decision cannot be changed at a later pipeline point. A COPY may be changed at a later point to either a REDIR or DISCARD action.

- If a copy is changed to a discard, the CPU still receives the frame copy.
- If a copy is changed to a redirection, the CPU receives a copy and the frame is looped back or also sent to the redirection queue.

When frames are copied or redirected the IFH is updated as described in the following table.

Table 173 • **Loopback IFH Updates**

IFH Field	Updated by
FWD.SRC_PORT	COPY or REDIR. The frame source port is changed to the current destination port.
MISC.PIPELINE_PT	COPY or REDIR. The pipeline point in which the frame forwarding was updated.
MISC.PIPELINE_ACT	COPY or REDIR. The pipeline action assigned to the frame.
MISC.CPU_MASK	Any operation. A CPU copy may be generated by all forward selections.
FWD.DO_LBK	Rewriter loopback. This bit is set to indicate that the frame has been looped by the REW. Can optionally be used to prevent multiple loops of the same frame.
FWD.UPDATE_FCS	VOP. Set by the VOP if a looped frame changed.
VSTAX.MISC.ISDX	VOP. The frame ISDX can be changed by the VOP.
VSTAX.QOS.DP	VOP. The DP level can be set to 0 by the VOP.
ENCAP.MPLS_TTL	VOP. The MPLS_TTL can be set to 1 by the VOP.

3.28.5.3 Egress Statistics Update

The rewriter does not have Egress Service Index (ESDX) and port-based counters of its own. Instead, the egress statistic counters in XQS:STAT are used for all frame and byte counting. For more information, see Statistics, page 306.

The following counters are available to the rewriter as frame and octet counters.

- Per egress port: 2 × 8 + 1 = 17 counters (color × cell bus priority + abort)
- ESDX: 2 × 1,024 = 2,048 counters (color × ESDX)

Frame and octets are counted simultaneously on each index by two separate counters. The counter widths are 40 bits for octet counters and 32 bits for frame counters.

The following table lists the registers associated with ESDX counter configuration.

Table 174 • **Egress Statistics Control Registers**

Register	Description	Replication
CNT_CTRL. STAT_MODE	Configures ESDX counter index selection.	Global
CNT_CTRL. VSTAX_STAT_ESDX_DIS	Configures ESDX counting for stacking ports. If this bit is set and PORT_CTRL.VSTAX_HDR_ENA = 1, all counting based on the ESDX value is disabled regardless of the CNT_CTRL.STAT_MODE configuration.	Global
CNT_CTRL. STAT_CNT_FRM_ABORT_ENA	Enables counting of frames discarded by the rewriter. This bit only enables counting of frames discarded by ES0 or the SW_MIP. Frame discards done by the VOP are not included.	Global

Table 174 • **Egress Statistics Control Registers *(continued)***

Register	Description	Replication
PORT_CTRL. XTR_STAT_PIPELINE_PT	Configures the extraction statistics pipeline point. Frames extracted before the configured pipeline point are not counted by the ESDX counters. Configuring the REW_IN_MIP value will cause all extracted frames to be counted. Default is REW_OU_SW.	Port
PORT_CTRL. INJ_STAT_PIPELINE_PT	Configures the injection statistics pipeline point. Frames injected after the configured pipeline point are not counted by the ESDX counters. Configuring the REW_VRAP value will cause all injected frames to be counted. Default is REW_OU_VOE.	Port

3.28.5.4 Statistics Counter Index Calculation

The port counter index is always the physical port number and priority. The frame color is determined in the same manner as the ESDX counter.

The ESDX counter index calculation is controlled by the CNT_CTRL.STAT_MODE register.

Table 175 • **ESDX Index Selection**

STAT_MODE	Description
0	Use ESDX if available: ESDX is available if ES0 lookup is enabled and if ES0.ESDX_BASE is different from 0. Index = ES0.ESDX_BASE + ES0.ESDX_COSID_OFFSET(IFH.VSTAX.MISC.COSID). If ESDX is not available: Index = IFH.VSTAX.MISC.ISDX (4,096 available. The two MSB bits are ignored.) Regardless of the index selection, the frame color is: Color = DP_MAP.DP(IFH.VSTAX.QOS.DP)
1	Similar to STAT_MODE = 0, except counting is disabled if the ESDX is not available.
2	Use ISDX: Index = IFH.VSTAX.MISC.ISDX. The frame color is used to count multicast frames: Color = FRAME.ETH_LINK_LAYER.DMAC (bit 40).
3	Use VID: Index = Classified VID (IFH.VSTAX.TAG.VID), if no tag is pushed Index = VID of outermost pushed tag, if a tag is pushed Index = REW:VMID:RLEG_CTRL.RLEG_EVID (IFH.DST..ERLEG) if L3. The frame color is used to count multicast frames: Color = FRAME.ETH_LINK_LAYER.DMAC (bit 40). The two MSB bits of the VID are ignored. 1,024 counter indexes are available.

Both port counters and ESDX counters are updated based on pipeline points and actions. For more information, see Rewriter Pipeline Handling, page 331. The ESDX statistics pipeline points can be configured using the PORT_CTRL.INJ_STAT_PIPELINE_PT and PORT_CTRL.XTR_STAT_PIPELINE_PT registers.

The following table shows when the counters are updated.

Table 176 • **Egress Statistics Update**

Counter	Is Updated When?
Port	Pipeline action is not Extract (action different from XTR, XTR_LATE_REW, XTR_UPMEP, and LBK_ASM). Frame is transmitted (abort = 0). Updated for all egress port numbers including virtual device (VD) and extraction CPU port numbers.
ESDX	The egress port is a physical port. The frame matches the parameters configured in CNT_CTRL.STAT_MODE: For extracted frames (Action: XTR, XTR_UPMEP and LBK_ASM): The egress statics is located in the pipeline point, PORT_CTRL.XTR_STAT_PIPELINE_PT. Frames extracted in or after this point are counted. For injected frames (Action: INJ, INJ_x, LBK_QS): The egress statics is located the pipeline point, PORT_CTRL.INJ_STAT_PIPELINE_PT. Frames injected in or before this point are counted. Frame is transmitted (abort = 0). The pipeline action is different from XTR_LATE_REW.
Abort	A frame is aborted by REW or OQS (abort = 1). Counting of frames aborted by REW must be enabled by setting CNT_CTRL.STAT_CNT_FRM_ABORT_ENA.

3.28.5.5 Protection Active

Using ES0, the rewriter is able to change various indexes based on the value of the IFH.DST.ENCAP.PROT_ACTIVE field in the IFH.

- Use the ES0 key field PROT_ACTIVE to change all ES0 actions when protection is active.
- The numbers of ES0 keys are limited, so the same key can optionally generate two different indexes based on the value of IFH.DST.ENCAP.PROT_ACTIVE. This is enabled using the ES0 action PROTECTION_SEL.
- The ENCAP_ID, ESDX_BASE and OAM_MEP_IDX indexes can be changed. These indexes are used when enabled by PROTECTION_SEL: ENCAP_ID_P, ESDX_BASE_P, and OAM_MEP_IDX_P.

3.28.6 Mapping Tables

The rewriter contains two mapping resources (A and B) that can be used to look up the following fields.

- TC value in MPLS labels
- DSCP value in IP frames
- DEI in 802.3 VLAN tags
- PCP in 802.3 VLAN tags

The following table lists the configuration registers associated with the mapping tables.

Table 177 • **Mapping Table Configuration Registers**

Register	Description	Replication
MAP_RES_x (x= A or B)		
MAP_VAL_x:TC_VAL	Mapped TC value	2,048 × 2
MAP_VAL_x:DSCP_VAL	Mapped DSCP value	2,048 × 2
MAP_VAL_x:DEI_VAL	Mapped DEI value	2,048 × 2
MAP_VAL_x:PCP_VAL	Mapped PCP value	2,048 × 2
MAP_VAL_x:OAM_COLOR	Mapped OAM Color	2,048 × 2
MAP_VAL_x:OAM_COSID	Mapped OAM Cosid	2,048 × 2

Table 177 • **Mapping Table Configuration Registers *(continued)***

Register	Description	Replication
MAP_LBL_x:LBL_VAL	Mapped MPLS label value	2,048 × 2

Each of the two resources provide all of the fields listed above. Two lookups are done in each resource for every frame. This generates four mapping results per frame.

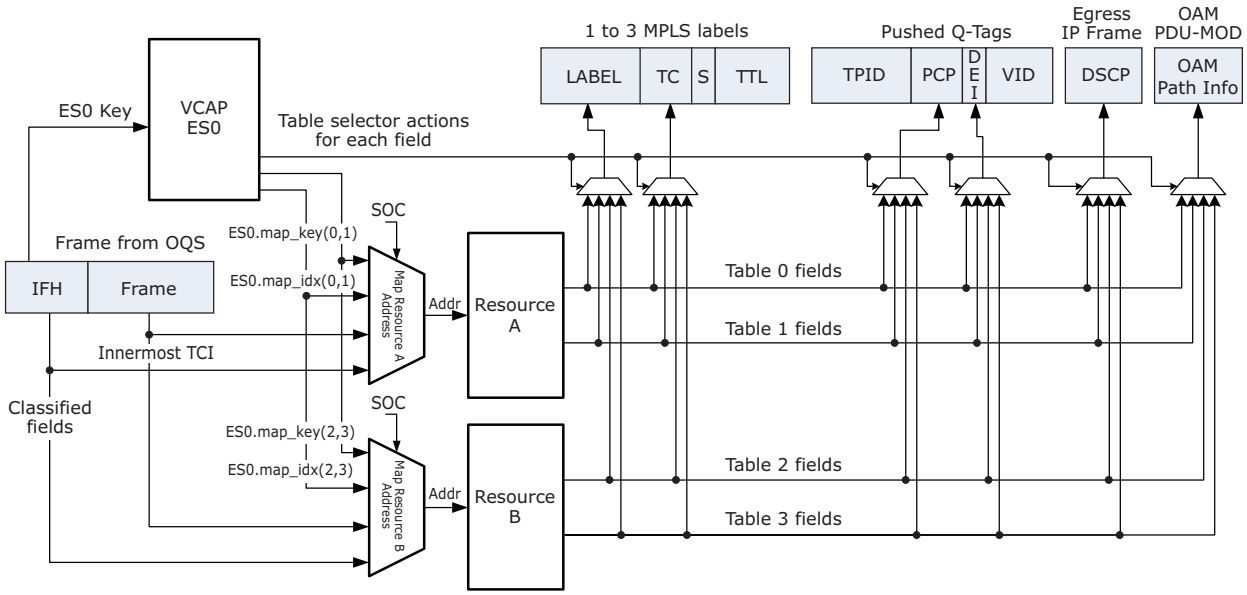
The four lookups implement mapping table 0 to 3. The address used for each lookup is controlled by the ES0 actions MAP_n_KEY and MAP_n_IDX (*n* = 0:3).

A mapping table address consists of two parts:

- A base index: ES0.MAP_n_IDX. This controls the MSB part of the table address.
- One of 15 different keys that use classified frame properties to generate the LSB of the table address. For more information about each key, see ES0 Configuration Registers, page 323.

The following illustration depicts the mapping table functionality.

Figure 114 • **Mapping Tables**



The mapping tables provide a flexible way to update frame fields. The following example shows how to configure and use mapping table 1.

Update PCP in VLAN tag A using mapping table 1. Use frame DSCP as index:

- Enable ES0
- Program an ES0 key
- Set ES0 action TAG_A_PCP_SEL to 5 for selected key
- Set ES0 action PUSH_OUTER_TAG = 1
- Update PCP by using Table 1:
  Select a base address in resource A. Base address 256 is used in this example
  Set ES0 action MAP_1_IDX = 256/8 = 32
- Use frame DSCP as key: Set ES0 action MAP_1_KEY = 2
- Program the PCP mapping in table 1: Addr = 256 to 256 + 63 (all DSCP values)
- Write REW:MAP_RES_A:$Addr/MAP_VAL_A PCP_VAL <New PCP>

Frame DSCP is used as key in this example. If the DSCP is not available (frame is non IP), the mapping is considered invalid, and the PCP value is selected using this fallback method:

- Use mapping table 1 if key is valid, otherwise use mapping table 0.
- Use mapping table 0 if key is valid, otherwise use a fixed value.

In the example, a PCP value from table 0 is used instead for non IP frames, if the table 0 key is valid. Otherwise, the ES0 action PCP_A_VAL is used. Mapping tables 2 and 3 have the same functionality.

Mapping tables 0 and 1 share the 2,048 addresses available in Resource A. Table 2 and 3 share 2,048 addresses in Resource B.

3.28.7 VLAN Editing

The rewriter performs five steps related to VLAN editing for each frame and destination:

1. ES0 lookup. ES0 is looked up for each frame if enabled by the global configuration. The action from ES0 controls the pushing of VLAN tags.
2. VLAN popping. Zero to three VLAN tags are popped from the frame. Popping is controlled from the ANA by IFH.TAGGING.POP_CNT and by the action ES0.POP_VAL.
3. VLAN push decision. Decides the number of new tags to push and which tag source to use for each tag. Tag sources are port, ES0 (tags A to C), and routing.
4. Constructs the VLAN tags. The new VLAN tags are constructed based on the tag sources' configuration.
5. VLAN pushing. The new VLAN tags are pushed.

The outermost tag can optionally be controlled by port-based VLAN configuration.

Table 178 • **Port VLAN Tag Configuration Registers**

Register	Description	Replication
TPID_CFG. TPID_VAL	Configures three custom TPID values. These must be configured identically in ANA_CL::VLAN_STAG_CFG.STAG_ETYPE_VAL.	3
PORT_VLAN_CFG PORT_*	Port VLAN TCI. Port _VID is also used for untagged set.	Ports
TAG_CTRL: TAG_CFG_OBEY_ WAS_TAGGED	Controls how port tags are added. If this bit is set, the IFH field VSTAX.TAG.WAS_TAGGED must be 1 before a port tag is added to a frame.	Ports
TAG_CTRL: TAG_CFG	Controls port tagging. Four modes are supported.	Ports
TAG_CTRL: TAG_TPID_CFG	Selects Tag Protocol Identifier (TPID) for port tagging.	Ports
TAG_CTRL: TAG_VID_CFG	Selects VID in port tag.	Ports
TAG_CTRL: TAG_PCP_CFG	Selects PCP fields in port tag.	Ports
TAG_CTRL: TAG_DEI_CFG	Selects DEI fields in port tag.	Ports
DP_MAP.DP	Maps the four drop precedence levels (DP) to a drop eligible value (DE or COLOR).	Global
PORT: PCP_MAP_DEx	Mapping table. Maps DP level and QoS class to new PCP values.	Per port per Q per DE
PORT: DEI_MAP_DEx	Mapping table. Maps DP level and QoS class to new DEI values.	Per port per Q per DE

3.28.7.1 ES0 Lookup

For each frame passing through the rewriter, an optional VCAP_ES0 lookup is done using the appropriate key. The action from ES0 is used in the following to determine the frame's VLAN editing.

3.28.7.2 VLAN Popping

The rewriter initially pops the number of VLAN tags specified by the IFH field IFH.TAGGING.POP_CNT received with the frame. Up to three VLAN tags can be popped via the IFH. The rewriter itself can further influence the number VLAN tags being popped by the ES0 action POP_VAL. This can increase the pop count by one.

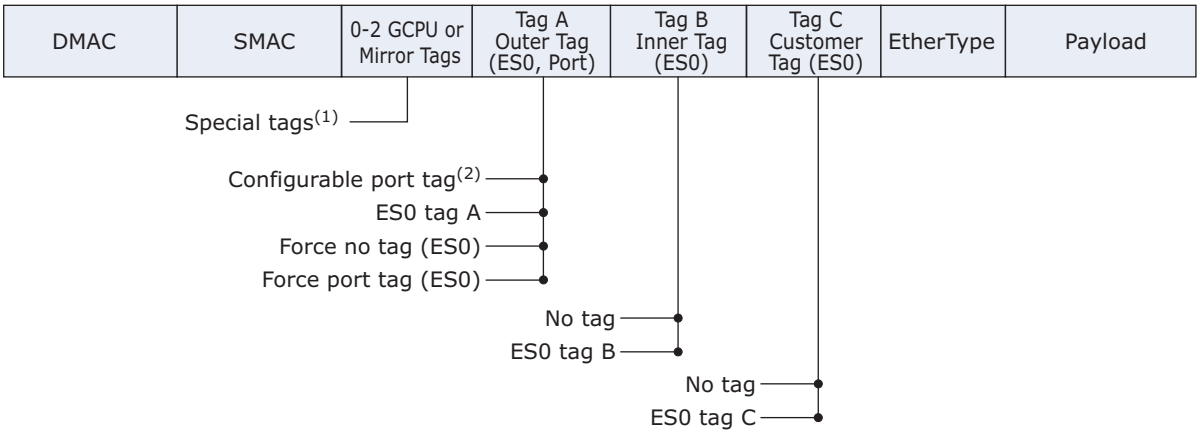
The rewriter analyzes the total number of tags in a frame, and this is the maximum number of tags that can be popped regardless of the IFH and ES0 pop values.

3.28.7.3 VLAN Push Decision

After popping the VLAN tags, the rewriter decides to push zero to three new VLAN tags into the outgoing frame according to the port's tagging configuration in register TAG_CTRL.TAG_CFG and the action from a potential VCAP ES0 hit.

Two additional tags can be added for mirrored or GCPU frames. These special tags will always be the outermost ones. If MPLS encapsulation is added to a frame, the mirror or GCPU tags are placed after the MPLS link layer SMAC.

Figure 115 • **VLAN Pushing**



1. Mirror tag is controlled by REW::MIRROR_PROBE_CFG.REMOTE_MIRROR_CFG
   GCPU tag is controlled by REW::GCPU_CFG.GCPU_TAG_SEL.
2. Port tag is controlled by REW:PORT:TAG_CTRL.TAG_CFG.

The following table lists ES0 VLAN tag actions.

Table 179 • **ES0 VLAN Tag Actions**

ES0_ACTION	TAG_CFG	Tagging Action
No ES0 hit	Controls port tag	The port tag is pushed according to TAG_CTRL.TAG_CFG as outermost tag. The following options are available: - Tag all frames with port tag. - Tag all frames with port tag, except if classified VID = 0 or classified VID=PORT_VLAN_CFG.PORT_VID. - Tag all frames with port tag, except if classified VID = 0. TAG_CTRL.TAG_VID_CFG selects between classified- or fixed-port VID. No inner tag.
PUSH_OUTER_TAG = 0 PUSH_INNER_TAG = 0	Controls port tag	Same as No ES0 hit action.
PUSH_OUTER_TAG = 1 PUSH_INNER_TAG = 0	Don't care	ES0 tag A is pushed as outer tag. No inner tag.

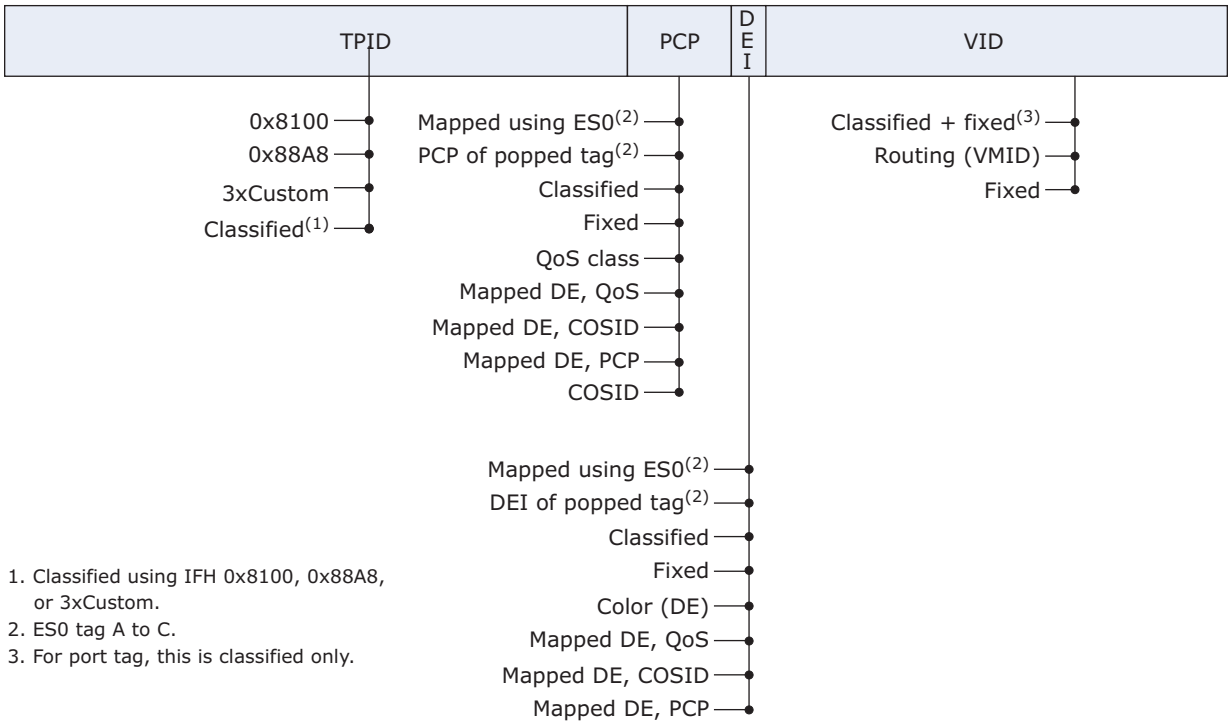
Table 179 • **ES0 VLAN Tag Actions *(continued)***

ES0_ACTION	TAG_CFG	Tagging Action
PUSH_OUTER_TAG = 2 PUSH_INNER_TAG = 0	Don't care	Port tag is pushed as outer tag. This overrules port settings in TAG_CFG. No inner tag.
PUSH_OUTER_TAG = 3 PUSH_INNER_TAG = 0	Don't care	No tags are pushed. This overrules port settings in TAG_CFG.
PUSH_OUTER_TAG = 0 PUSH_INNER_TAG = 1	Controls port tag	Port tag is pushed according to TAG_CFG as outermost tag. The following options are available: - Tag all frames with port tag. - Tag all frames with port tag, except if classified VID = 0 or classified VID = PORT_TAG_DEFAULT. PORT_TCI_VID. - Tag all frames with port tag, except if classified VID = 0. - No port tag. TAG_CTRL.TAG_VID_CFG select between classified- or fixed-port VID. ES0 tag B is pushed as inner tag. ES0 tag B is effectively the outer tag if the port tag is not pushed.
PUSH_OUTER_TAG = 1 PUSH_INNER_TAG = 1	Don't care	ES0 tag A is pushed as outer tag. ES0 tag B is pushed as inner tag.
PUSH_OUTER_TAG = 2 PUSH_INNER_TAG = 1	Don't care	Port tag is pushed as outer tag. This overrules port settings in TAG_CFG. ES0 tag B is pushed as inner tag.
PUSH_OUTER_TAG = 3 PUSH_INNER_TAG = 1	Don't care	No outer tag is pushed. This overrules port settings in TAG_CFG. ES0 tag B is pushed as inner tag. ES0 tag B is effectively the outer tag, because no outer tag is pushed.
Tag C is controlled separately and will always be the inner most tag. All combinations of tags A to C are allowed.		
PUSH_CUSTOMER_TAG = 0	Controls port tag	Does not push customer tag (tag C).
PUSH_CUSTOMER_TAG = 1	Controls port tag	Push tag C if IFH.VSTAX.TAG.WAS_TAGGED = 1.
PUSH_CUSTOMER_TAG = 2	Controls port tag	Push tag C if IFH.VSTAX.TAG.WAS_TAGGED = 0.
PUSH_CUSTOMER_TAG = 3	Controls port tag	3: Push tag C if UNTAG_VID_ENA = 0 or (C-TAG.VID ! = VID_C_VAL). C-TAG.VID is controlled by TAG_C_VID_SEL.
UNTAG_VID_ENA	Controls port tag	See PUSH_CUSTOMER_TAG = 3.

3.28.7.4 Constructing VLAN Tags

The contents of the tag header are highly programmable when pushing a VLAN tag. The following illustration shows the VLAN tag construction.

Figure 116 **• VLAN Tag Construction**



The port tags (ES0.PUSH_OUTER_TAG = [0[2]), the ES0 tags A to C, and the special tags have individual configurations.

Port Tag: PCP

Use REW:PORT:TAG_CTRL.TAG_PCP_CFG to configure the following:

- Use the classified PCP.
- Use the egress port's port VLAN (PORT_VLAN_CFG.PORT_PCP).
- Use the QoS class directly.
- Map DE and QoS class to a new PCP value using the per-port table PCP_MAP_DEx.
- Map DE and COSID to a new PCP value using the per-port table PCP_MAP_DEx.
- Map DE and classified PCP to a new PCP value using the per-port table PCP_MAP_DEx.
- Use the COSID class directly.

Port Tag: DEI

Use REW:PORT_TAG_CTRL.TAG_DEI_CFG to configure the following:

- Use the classified DEI.
- Use the egress port's port VLAN (PORT_VLAN_CFG.PORT_DEI).
- Use mapped DP to DE level (color).
- Map DE and QoS class to a new DEI value using the per-port table DEI_MAP_DEx.
- Map DE and COSID to a new DEI value using the per-port table DEI_MAP_DEx.
- Map DE and classified PCP to a new DEI value using the per-port table DEI_MAP_DEx.

Port Tag: VID

Use REW:PORT:TAG_CTRL.TAG_VID_CFG to configure the following:

- Use the classified VID.
- Use the egress port's port VLAN (PORT_VLAN_CFG.PORT_VID).
- Use VMID from router leg lookup. When L3 routing is active, the classified VID is overwritten by the VMID.

Port Tag: TPID

Use REW:PORT:TAG_CTRL.TAG_TPID_CFG to configure the following:

- Use Ethernet type 0x8100 (C-tag).
- Use Ethernet type 0x88A8 (S-tag).
- Use one of three custom TPIDs programmed in REW::TPID_CFG.TPID_VAL.
- The TPID is classified by ANA and selected by IFH.

Similar options for the ES0 tag A to tag C are available:

ES0 Tag: PCP

Use ES0 action TAG_x_PCP_SEL to configure the following:

- Use the classified PCP.
- Use ES0_ACTION.PCP_x_VAL.
- Use PCP of popped tag (IFH.VSTAX.TAG.WAS_TAGGED = 1 and num_popped_tags>0) else PCP_x_VAL.
- Use the complex mapping functionality provided by ES0 to lookup a PCP value.

ES0 Tag: DEI

Use ES0 action TAG_x_DEI_SEL to configure the following:

- Use the classified DEI.
- Use ES0_ACTION.DEI_x_VAL.
- Use the complex mapping functionality provided by ES0 to lookup a DEI value.
- Use DEI of popped tag (IFH.VSTAX.TAG.WAS_TAGGED = 1 and num_popped_tags>0) else DEI_x_VAL.
- Use the mapped DP to DE level (color) directly.

ES0 Tag: VID

Use ES0 action TAG_x_VID_SEL to configure the following:

- Tag A and B: Use the classified VID incremented with ES0.VID_x_VAL.
  Tag C: Use the classified VID.
- Use ES0_ACTION.VID_x_VAL.

ES0 Tag: TPID

Use ES0 action TAG_x_TPID_SEL to configure the following:

- Use Ethernet type 0x8100 (C-tag).
- Use Ethernet type 0x88A8 (S-tag).
- Use one of three custom TPIDs programmed in REW::TPID_CFG.TPID_VAL.
- TPID is classified by ANA and selected via IFH.

GCPU or mirror forwarding can optionally insert additional tags. These will always be added as the outermost tags. One or two special tags (A and B) can be added. The tag construction options as follows:

Special Tag A+B: PCP

- Use the classified PCP.
- Use fixed value.

Special Tag A+B: DEI

- Use fixed value.

Special Tag A+B: VID

- Use fixed value.

Special Tag A+B: TPID

- Use Ethernet type 0x8100 (C-tag).
- Use Ethernet type 0x88A8 (S-tag).
- Use one of three custom TPIDs programmed in REW::TPID_CFG.TPID_VAL.
- TPID is classified by ANA and selected by means of IFH.

3.28.8 DSCP Remarking

The rewriter supports two DSCP remarking schemes. The first remarking mode is controlled by port and IFH settings. The second remarking mode is controlled by ES0 actions and by the general mapping tables.

DSCP actions from ES0 override any remarking done by the port.

3.28.8.1 ES0-Based DSCP Remarking

ES0 controls DSCP remarking if ES0 lookups are enabled and the DSCP_SEL action is different from 0. For more information, see Mapping Tables, page 336.

3.28.8.2 Legacy Port-Based Mode

Simple DSCP remapping can be done using a common table shared by all ports.

The following table shows the port-based DSCP editing registers associated with remarking.

Table 180 • **Port-Based DSCP Editing Registers**

Register	Description	Replication
DP_MAP.DP	Maps the four drop precedence levels to a drop eligible value (DE). DE is also called color.	Global
DSCP_REMAP.DSCP_REMAP	Full one-to one DSCP remapping table common for all ports.	None
DSCP_MAP.DSCP_UPDATE_ENA	Selects DSCP from either frame or IFH.QOS.DSCP. If DSCP_MAP.DSCP_UPDATE_ENA = 1 and IFH.QOS.UPDATE_DSCP = 1: Selected DSCP = IFH.QOS.DSCP Else keep frame DSCP: Selected DSCP = Frame DSCP.	Ports
DSCP_MAP.DSCP_REMAP_ENA	Optionally remaps selected DSCP. If DSCP_MAP.DSCP_REMAP_ENA = 1 and IFH.QOS.TRANSPARENT_DSCP = 0: New DSCP = DSCP_REMAP [Selected DSCP]. Else do no remap: New DSCP = selected DSCP.	Ports

The following remarking options are available.

- No DSCP remarking. The DSCP value in the frame is untouched.
- Update the DSCP value in the frame with the value received from the analyzer in IFH.QOS.DSCP.
- Update the DSCP value in the frame with the frame DSCP mapped through DSCP_REMAP.DSCP_REMAP.
- Update the DSCP value in the frame with the value received from the analyzer (IFH.QOS.DSCP) mapped through DSCP_REMAP.DSCP_REMAP.

ANA can set IFH.QOS.TRANSPARENT_DSCP to prevent DSCP mapping even if this is enabled for a port. ANA can also force use of the frame DSCP for mapping by setting IFH.QOS.UPDATE_DSCP to 0.

The IP checksum is updated when changing the DSCP for IPv4 frames.

3.28.9 VStaX Header Insertion

The rewriter controls insertion of the VStaX header. For more information about the header fields, see VStaX Header, page 29.

The following registers are used for configuration of the rewriter specific stacking functionality.

Table 181 • **Rewriter VStaX Configuration Registers**

Register	Description	Replication
COMMON_CTRL. OWN_UPSID	Configures own UPSID to be used for stacking.	None

Table 181 • **Rewriter VStaX Configuration Registers *(continued)***

Register	Description	Replication
CNT_CTRL. VSTAX_STAT_ESDX_DIS	Configures ESDX counting for stacking ports. If this bit is set and PORT_CTRL.VSTAX_HDR_ENA = 1, all counting based on the ESDX value is disabled regardless of the CNT_CTRL.STAT_MODE configuration.	None
PORT_CTRL. VSTAX_STACK_GRP_SEL	Selects logical stacking port (or stacking port group) membership.	Ports
PORT_CTRL. VSTAX_HDR_ENA	Enables insertion of stacking header in frame.	Ports
PORT_CTRL. VSTAX_PAD_ENA	Enables padding of frames to 76 bytes. Setting this bit will cause all frames on the port to be extended to 76 bytes instead of 64 bytes. This should only optionally be enabled for stacking ports (PORT_CTRL.VSTAX_HDR_ENA = 1). Setting this bit will prevent frames from becoming under sized in a receiving switch, when the VStaX header is removed. See ASM::ERR_STICKY.FRAGMENT_STICKY.	Ports
PORT_CTRL.VSTAX2_MIRROR_OBEY_WAS_TAGGED	Configures tagging of frames with VSTAX.GEN.FWD_MODE = VS2_FWD_GMIRROR. Only active on front ports for frames with this FWD_MODE. This is used to control the remote mirror tagging of frames that have been mirrored from one unit in the stack to another unit.	Ports
PORT_CTRL. VSTAX2_MISC_ISDX_ENA	Configures VSTAX MISC field decoding. Select between MISC- or AC mode.	Ports
VSTAX_PORT_GRP_CFG. VSTAX_TTL	Link TTL value.	2
VSTAX_PORT_GRP_CFG. VSTAX_LRN_ALL_HP_ENA	Changes priority of learn all frames to the highest priority (IFH.VSTAX.QOS = 7).	2
VSTAX_PORT_GRP_CFG. VSTAX_MODE	Controls whether forwarding modes specific to VStaX AF are translated to BF forwarding modes. If set to 0, the following translation is performed: fwd_logical -> fwd_llookup fwd_mc -> fwd_llookup If set to 1, no translation is performed. Translation is only required for advanced forwarding switches operating in a basic forwarding stack.	2

3.28.10 Forwarding to GCPU

The OQS can redirect CPU frames on selected CPU queues to the external ports instead of sending them to the internal CPU. This is called GCPU forwarding and is typically used if an external CPU is connected to the VSC7430 device through an Ethernet port.

GCPU forwarding is controlled per individual CPU queue. Stack ports have additional options for GCPU forwarded frames.

Table 182 • **GCPU Configuration Registers**

Register	Description	Replication
GCPU_CFG. GCPU_KEEP_IFH	Used when GCPU frames are forwarded to a front port. Frames are sent with the IFH preserved. The IFH is encapsulated according to the configuration. Setting GCPU_KEEP_IFH to a value different from 0 overrides the GCPU_TAG_SEL and GCPU_DO_NOT_REW settings for front ports. No other rewrites are done to the frame. The GCPU_KEEP_IFH setting is not active if PORT_CTRL.KEEP_IFH_SEL is different from 0 or if PORT_CTRL.VSTAX_HDR_ENA = 1.	Per CPU Q
GCPU_CFG. GCPU_DO_NOT_REW	Used when GCPU frames are forwarded to a front port. Except for the optional tags configured in GCPU_CFG.GCPU_TAG_SEL, frames are not modified when forwarded to the GCPU.	Per CPU Q
GCPU_CFG. GCPU_TAG_SEL	The destination port type determines the functionality. When a GCPU frame is forwarded to a stack port: Force a change of the VSTAX.TAG to the configured values in GCPU_TAG_CFG:0. When a GCPU frame is forwarded to a front port: Optionally add one or two Q-tags to the frame. The tags are configured using GCPU_TAG_CFG.	Per CPU Q
GCPU_CFG. GCPU_FWD_MODE	Used when GCPU frames are forwarded to a stack port. Controls the value of VSTAX.DST.DST_PN.	Per CPU Q
CPU_CFG. GCPU_UPSPN	Used when GCPU frames are forwarded to a stack port. CPU QUEUE to be used as destination queue for global CPU transmission. The value depends on configuration and the value of GCPU_CFG.GCPU_FWD_MODE. Controls the value of VSTAX.DST.DST_PN.	Per CPU Q
GCPU_CFG. GCPU_UPSID	Used when GCPU frames are forwarded to a stack port. UPSID to be used as destination in the VStaX header. Sets VSTAX.DST.DST_UPSID to configured value.	Per CPU Q
GCPU_TAG_CFG. TAG_TPID_SEL	Configuration of Q-Tags for GCPU frames. GCPU frames that are forwarded to a front port can optionally have one or two IEEE 802.3 VLAN tags added. The tags will be placed in the outer most position after the SMAC. GCPU VLAN tag Protocol Identifiers (TPID).	2
GCPU_TAG_CFG. TAG_VID_VAL	GCPU VLAN VID value.	2
GCPU_TAG_CFG. TAG_DEI_VAL	GCPU VLAN DEI values.	2
GCPU_TAG_CFG. TAG_PCP_SEL	Selection of GCPU VLAN PCP values.	2
GCPU_TAG_CFG. TAG_PCP_VAL	GCPU VLAN PCP values.	2

The IFH.MISC.CPU_MASK field is used to select the GCPU forwarding configuration. If multiple bits are set in this mask, the most significant bit is always used to control the GCPU forwarding. The ANA_AC::CPU_CFG.ONE_CPU_COPY_ONLY_MASK register is used to control how the CPU_MASK is set before the frame enters the rewriter. If the CPU_MASK field is 0, all GCPU forwarding is disabled.

GCPU frame forwarding must be enabled in the OQS per CPU queue using the QFWD::FRAME_COPY_CFG.FRMC_PORT_VAL registers.

3.28.11 Layer 3 Routing

The following registers are associated with routing (Layer 3, or L3). They are used when the analyzer signals routing by setting IFH.FWD.DST_MODE = L3UC_ROUTING or L3MC_ROUTING.

The analyzer can also signal routing using IFH.DST.ENCAP.RT_FWD = 1. In this case, the MAC addresses and VMIDs are controlled directly by the analyzer, and the registers in the rewriter are not used.

Table 183 • **Routing SMAC Configuration Registers**

Register	Description	Replication
RLEG_CFG_0. RLEG_MAC_LSB	Router Leg SMAC address used for IP routing (least significant bits). Must be set identical to ANA_L3::RLEG_CFG_0.RLEG_MAC_LSB.	None
RLEG_CFG_1. RLEG_MAC_TYPE_SEL	Configures how Router Leg MAC addresses are specified. Must be set identical to ANA_L3::RLEG_CFG_1.RLEG_MAC_TYPE_SEL. 0: RLEG_MAC:= RLEG_MAC_MSB(23:0) and (RLEG_MAC_LSB(23:0) + VMID(7:0) mod 2^{24}) 1: RLEG_MAC:= RLEG_MAC_MSB(23:0) and (RLEG_MAC_LSB(23:0) + VID(11:0) mod 2^{24} Others: RLEG_MAC:= RLEG_MAC_MSB(23:0) and RLEG_MAC_LSB(23:0) number of common address for all VLANs.	None
RLEG_CFG_1. RLEG_MAC_MSB	MSB part of SMAC used for IP routing. Must be set identical to ANA_L3::RLEG_CFG_1.RLEG_MAC_MSB.	None

The IFH.DST.L3_[MC|UC]ROUTING.ERLEG fields are used to select the VID of routed frames using the Egress Mapped VLAN (VMID) configuration registers listed in the following table.

Table 184 • **L3 Routing Egress Mapping VLAN (EVMID)**

Register	Description	Replication
RLEG_CTRL.RLEG_EVID	VID value assigned to this router leg.	32
RLEG_CTRL. RLEG_VSTAX2_WAS_TAGGED	Controls the value of IFH.VSTAX.TAG.WAS_TAGGED field in the stack header for frames that are L3 forwarded to a stack port.	32

3.28.12 OAM Frame Handling

The main OAM functionality is handled by the central VOP block. For every OAM frame coming through the rewriter, the OAM Protocol Data Unit (PDU) is forwarded to the VOP, and a response is sent back to REW when the VOP completes the PDU processing.

The REW includes the generic OAM PDU-MOD block that is controlled directly by the VOP response. The PDU-MOD block performs the required OAM PDU rewrites and handles OAM egress statistics.

The VOP can change fields in the IFH including frame ISDX for loopback frames. For more information about the VOP and PDU-MOD functionality, see Internal Frame Header Layout, page 23.

The following table lists the rewriter configuration registers that control the selection of outer VLAN PCP and DEI values, which are used by the VOP to generate counter indexes.

Table 185 • **Rewriter OAM Outer TCI Registers**

Register	Description	Replication
PORT_CTRL. PORT_VOE_TPID_AWARE_DIS	This configuration applies to VLAN tag awareness in the port VOE for frames for which the rewriter is not pushing new VLAN tags or an MPLS link layer. Each bit corresponds to one of the known TPIDs. If the outgoing frame's outer tag contains a TPID for which PVE_TPID_AWARE_DIS is set, then the port VOE sees the frame as untagged.	Ports
PORT_CTRL. PORT_VOE_DEFAULT_PCP	Default PCP value used by the OAM port VOE. This value is used for port VOE counter updates when no outer Q-tag is present in a frame.	Ports
PORT_CTRL. PORT_VOE_DEFAULT_DEI	Default DEI value used by the OAM port VOE. This value is used for port VOE counter updates when no outer Q-tag is present in a frame.	Ports

3.28.12.1 Y.1731 Ethernet MIP

The rewriter contains a MIP table with 128 entries. Each of these entries can be used to create an Up-MIP half function. Down-MIP half functions are created by similar functionality in the analyzer.

The task of the MIP is to extract relevant OAM PDUs from the frame flow for further MIP software processing. The MIP reacts to its own MEL; it does not filter frames with other MELs. The MIP handles the OAM functions CCM, LBM, LTM, and RAPS. In addition, one generic OAM function can be configured.

The OAM MIP is activated by assigning a MIP entry to frames using the ES0 action MIP_IDX. The MIP is active if the MIP_IDX is different for 0 and if the frame is one of the following Y.1731 OAM PDUs types:

- CCM
- LBM
- LTM
- Ring_APS
- Generic type

The analyzer is used to determine if a frame contains an Y.1731 PDU. The REW uses the IFH field IFH.DST.ENCAP.PDU_TYPE to determine the PDU type. The PDU_TYPE must be OAM_Y1731 before the MIP is active. For information about how to configure VCAP CLM, see Y.1731 Ethernet MIP, page 129.

The resulting action, if both the VOP and OAM MIP are active, is determined by the pipeline point assigned to the MIP. MIP actions can also be affected by the SW_MEP actions from ES0.

Each MIP entry can optionally be disabled by a VLAN tag VID filter. The VID filter can be configured to allow untagged and/or single tagged frames. The VID filter is configured by the MIP_VID_CTRL.VID_SEL setting.

A frame is singled tagged if the following conditions are met:

- One VLAN must be present in the frame after all VLAN pop operations have been applied and the TPID of that VLAN tag must be known (not disabled by PORT_CTRL.PORT_VOE_TPID_AWARE_DIS). The VID of this remaining tag is used by the VID filter.
- The frame is untagged or all of the tags are popped from the frame and the REW pushes one new VLAN tag. The VID of this pushed tag is used by the VID filter.
- Number of frame tags. Number of popped tags + number of pushed tags = 1

Note that VIDs of VLAN tags added by MPLS encapsulation cannot be used by the VID filter in the MIP. The VID must be present in the frame or be placed in a VLAN tag pushed by either the port tag configuration by an ES0 tag action.

The following table lists the configuration registers associated with the 128 MIPs.

Table 186 • **Y.1731 OAM MIP Registers**

Register	Description	Replication
REW:MIP_TBL:MIP_CFG. MEL_VAL	MEL value for the MIP.	128
REW:MIP_TBL:MIP_CFG. CCM_COPY_ENA	If set, OAM Y.1731 CCM frames with the correct encapsulation and the correct MEL are copied to the CPU.	128
REW:MIP_TBL:MIP_CFG. LBM_REDIR_ENA	If set, OAM Y.1731 LBM frames with the correct encapsulation, the correct MEL, and the correct destination MAC address are redirected to the CPU.	128
REW:MIP_TBL:MIP_CFG. LTM_REDIR_ENA	If set, OAM Y.1731 LTM frames with the correct encapsulation and the correct MEL are redirected to the CPU.	128
REW:MIP_TBL:MIP_CFG. RAPS_CFG	Handles OAM Y.1731 frames with Opcode = RAPS, correct encapsulation, and correct MEL.	128
REW:MIP_TBL:MIP_CFG. GENERIC_OPCODE_VAL	Generic Opcode. See GENERIC_OPCODE_CFG.	128
REW:MIP_TBL:MIP_CFG. GENERIC_OPCODE_CFG	Handles OAM Y.1731 frames with Opcode = GENERIC_OPCODE_VAL, correct encapsulation, and correct MEL.	128
REW:MIP_TBL:MIP_CFG. CPU_MIP_QU	CPU extraction queue when frame is forwarded to CPU.	128
REW:MIP_TBL:MIP_CFG. PIPELINE_PT	MIP pipeline point. 0: REW_IN_MIP 1: REW_OU_MIP	128
REW::MIP_CTRL. MIP_CCM_HMO_SET_SHOT	Sets all CCM Hit-Me-Once bits. Cleared when the access completes.	None
REW::MIP_CTRL. MIP_CCM_INTERVAL_MASK	Specifies which MIP CCM intervals that will have CCM_COPY_ONCE_ENA set. x0x: Interval is ignored x1x: REW:MIP_TBL:CCM_HMO_CTRL.CCM_COPY_ONCE_ENA is set where MIP_CCM_INTERVAL_MASK[CCM_HMO_CTRL.CCM_INTERVAL] is set.	None
REW:MIP_TBL: CCM_HMO_CTRL. CCM_INTERVAL	Interval used for automatically setting CCM_COPY_ONCE_ENA based on REW::MIP_CTRL.MIP_CCM_HMO_SET_SHOT. CCM_COPY_ONCE_ENA is set by hardware only if MIP_CCM_INTERVAL_MASK[CCM_HMO_CTRL.CCM_INTERVAL] is set.	128
REW:MIP_TBL: CCM_HMO_CTRL. CCM_COPY_ONCE_ENA	Sends the next received CCM frame to CPU. Cleared by hardware when a CPU copy is sent to CPU.	128
REW:MIP_TBL:MIP_VID_CTRL. VID_VAL	Required outer VID to identify frame as MIP.	128

Table 186 • **Y.1731 OAM MIP Registers *(continued)***

Register	Description	Replication
REW:MIP_TBL:MIP_VID_CTRL.VID_SEL	Outer VID check. Configure how the outer frame VID is matched before a frame is accepted as MIP. 0: VID check is disabled. Frame is always accepted. 1: Accept untagged frames. Tagged frames are not accepted. 2: Accept single tagged frames with VID = VID_VAL. Untagged frames or frames with multiple VLAN tags are not accepted. 3: Accept untagged frames and single tagged frames with VID = VID_VAL. Frames with multiple VLAN tags are not accepted.	128
REW:MIP_TBL:LBM_MAC_HIGH.LBM_MAC_HIGH	Destination MAC address bits 47:32 used for LBM. If LBM_MAC_HIGH = 0 and LBM_MAC_LOW = 0, the MAC address check for LBM frames is disabled.	128
REW:MIP_TBL:LBM_MAC_LOW.LBM_MAC_LOW	Destination MAC address bit 31:0 used for LBM. See LBM_MAC_HIGH.	128
MIP_STICKY_EVENT.MIP_CCM_COPY_STICKY	This bit is set if a CCM CPU has been copied to CPU.	None
MIP_STICKY_EVENT.MIP_LBM_REDIR_STICKY	This bit is set if a LBM PDU was redirected to the CPU.	None
MIP_STICKY_EVENT.MIP_LTM_REDIR_STICKY	This bit is set if a LTM PDU was redirected to the CPU.	None
MIP_STICKY_EVENT.MIP_RAPS_STICKY	This bit is set if a Ring APS PDU was handled.	None
MIP_STICKY_EVENT.MIP_GENERIC_STICKY	This bit is set if a generic PDU was handled.	None
MIP_STICKY_EVENT.MIP_LBM_DA_CHK_FAIl_STICKY	This bit is set if a destination MAC address check failed for LBM frame.	None
MIP_STICKY_EVENT.MIP_MEL_CHK_FAIL_STICKY	This bit is set if a MEL check failed for enabled OAM frames.	None

CCM frames can optionally be filtered using the Hit-Me-Once (HMO) functionality available to each MIP entry. The CCM_HMO_CTRL.CCM_COPY_ONCE_ENA bit can be used to do a onetime redirection of CCM frames. When a CCM frame hits a MIP entry where the CCM_COPY_ONCE_ENA is set, the bit will be cleared automatically, and no further CCM frames will be redirected until the bit is set again.

The CPU can initiate that the CCM_COPY_ONCE_ENA bit is set for all MIP entries by setting the MIP_CCM_HMO_SET_SHOT bit. This will update the CCM_COPY_ONCE_ENA bit if the interval is enabled for the MIP entry by the CCM_HMO_CTRL.CCM_INTERVAL value. Four independent update intervals are supported by the MIP_CCM_INTERVAL_MASK register.

3.28.12.2 Automatic Frame Field Updates for AFI Frame Flows

Frame flows injected by the AFI can be automatically updated by the REW to support multiple COSIDs for a single AFI flow. The new COSID is selected in a circular order from a set of enabled vales. The ISDX table keeps the current state of the COSID updates per ISDX value.

The multi COSID automatic update function is active if the following IFH fields are set by the AFI in the injected frames:

- IFH.FWD.TS_MODE = INJ_LBK
- IFH.TS.INJ_LBK.COS_NXT_SEL
  Selects one of three COS_NXT pointers per ISDX.(0:disable). See REW::ISDX_TBL:COS_CTRL.COS_NXT.

- IFH.TS.INJ_LBK.COS_MASK[7:0]
  Specifies the COSIDs to be used. Setting a bit to 1 enables the corresponding COSID. For example, a mask value of 0x0B enables COSID 0, 1 and 3. COS_MASK = 0 disables auto updates.
- IFH.TS.INJ_LBK.CHG_COSID_ENA
  Controls updating of IFH.VSTAX.MISC.COSID
- IFH.TS.INJ_LBK.CHG_OUTER_PCP_ENA
  Controls updating of outermost VLAN tag PCP value
- IFH.TS.INJ_LBK.CHG_IFH_TC_ENA
  Controls updating of IFH.DST.ENCAP.MPLS_TC
- IFH.TS.INJ_LBK.CHG_IFH_PCP_ENA
  Controls updating of IFH.VSTAX.TAG.UPRIO

The multi COSID function operates differently for injected Up-MEP or Down-MEP frames.

Up-MEP frames are injected by the AFI on the VD1 port and looped back to the ANA. Injected Up-MEP frames are modified when they pass through the REW the first time on the VD1 port. The IFH of the looped frames will be modified if enabled by the CHG-fields. The PCP of the outer most VLAN tag in the ETH link layer is changed if this is enabled. The IFH.TS.INJ_LBK.COS_NXT_SEL field is set to 0 to in the frame. This disables further COSID updates when the frame reaches the REW again after the loop back.

Down-MEP frames are injected by the AFI on a physical port. If enabled by the INJ_LBK.CHG bits the REW will use new COSID value for the selected fields. The INJ_LBK.CHG_OUTER_PCP_ENA field has no functionality in Down-MEP mode. The outer PCP value will instead be controlled by the normal VLAN tagging configuration.

The following table shows the configurations available per ISDX.

Table 187 • **ISDX Table Configuration Registers**

Register	Description	Replication

ISDX_TBL:COS_CTRL.COS_NXT	The auto updated COSID value is determined according to the following algorithm:
	mask = IFH.TS.INJ_LBK.COS_MASK[7:0]
	isdx = IFH.VSTAX.MISC.ISDX
	cos_nxt_sel = IFH.TS.INJ_LBK.COS_NXT_SEL
	if (cos_nxt_sel > 0 and isdx > 0 and mask > 0) {
	cos_nxt = REW:ISDX_TBL:COS_CTRL[IFH.VSTAX.MISC.ISDX].COS_NXT.[cos_nxt_sel-1]
	
	# Use cos_nxt to find next bit in cos_mask
	for idx in 0:7 {
	if (mask[(idx+cos_nxt) mod 8] = '1') {
	cosid_new = idx
	break
	}
	}
	# Update next pointer
	REW:ISDX_TBL:COS_CTRL[IFH.VSTAX.MISC.ISDX].COS_NXT[cos_nxt_sel-1] = ((cosid_new+1) mod 8)
	}
		3 × 512

3.28.13 Mirror Frames

The device offers three mirror probe sets that can be individually configured. The following table lists the rewriter configuration registers associated with mirroring.

Table 188 • **Mirror Probe Configuration Registers**

Register	Description	Replication
MIRROR_PROBE_CFG.MIRROR_TX_PORT	The Tx port for each mirror probe (from where rewrite configuration was taken).	Mirror probes
MIRROR_PROBE_CFG.REMOTE_ENCAP_ID	Selects encapsulation ID to use for remote mirror frames.	Mirror probes
MIRROR_PROBE_CFG.REMOTE_MIRROR_CFG	Enables encapsulation of mirrored frames. One or two Q-tags (Q-in-Q) or the encapsulation table can be used. In tag mode, the VLAN tags are added after the outer SMAC. In other words, if an MPLS link layer is added to the frame, the tags are added after the LL-SMAC. In encapsulation mode, an entry in the ENCAP table is used for encapsulation. This overrides any encapsulation selected by ES0 for the frame.	
MIRROR_TAG_x_CFG:TAG_x_PCP_VAL	Mirror Q-tag PCP value.	2× mirror probes
MIRROR_TAG_x_CFG:TAG_x_PCP_SEL	Selection of mirror Q-tag PCP values.	2× mirror probes
MIRROR_TAG_x_CFG:TAG_x_DEI_VAL	Mirror Q-tag DEI values.	2× mirror probes
MIRROR_TAG_x_CFG:TAG_x_VID_VAL	Mirror Q-tag VID values.	2× mirror probes
MIRROR_TAG_x_CFG:TAG_x_TPID_SEL	The tag protocol identifier (TPID) for the remote mirror VLAN tag.	2× mirror probes

The device supports mirroring frames from either Rx ports (ingress mirror) or Tx ports (egress mirror). Frames either received on a port (Rx mirror) or frames sent to a port (Tx mirror) are copied to a mirror destination port.

For Tx mirroring, the rewriter rewrites the frame sent to the mirror destination port, exactly as the original frame copy is rewritten when forwarded to the mirror probe port. The mirror probe port number must be configured in the rewriter using register MIRROR_TX_PORT.

For Rx mirroring, the rewriter makes no modifications to the frame, so an exact copy of the original frame as received on the mirror probe port (ingress port) is sent out on the mirror destination port.

The analyzer controls both Rx and Tx mirroring. For more information, see Mirroring, page 218. The rewriter obtains mirror information for the frame from the internal frame header. IFH.FWD.MIRROR_PROBE indicates if the frame is a mirror copy and if so, to which of three mirror probes the frame belongs. Encoding is as follows.

- MIRROR_PROBE = 0: Frame is not a mirror copy
- MIRROR_PROBE = 1: Frame is mirrored using mirror probe set 1
- MIRROR_PROBE = 2: Frame is mirrored using mirror probe set 2
- MIRROR_PROBE = 3: Frame is mirrored using mirror probe set 3

In addition, the IFH.FWD.RX_MIRROR bit indicates if the frame is Rx mirrored (1) or Tx mirrored (0).

When mirroring traffic to a device that is not directly attached to the VSC7430 device, it may be required to push one or two VLAN tags to mirrored frame copies as to not violate general VLAN membership for the mirror destination port. These VLAN tags are called “remote mirror VLAN tags” and can be configured individually for each mirror probe. When enabled, the remote mirror VLAN tags are pushed

onto all mirrored traffic by the rewriter whether it is an Rx mirrored frame or a Tx mirrored frame. For remote mirroring it is also possible to add MPLS capsulation using the encapsulation table if this is required for forwarding the frame.

The REMOTE_MIRROR_ENA register configures whether to add one or two remote mirror VLAN tags or MPLS encapsulation. The REMOTE_ENCAP_ID bit selects the MPLS encapsulation to be used.

For Tx mirroring, care must be taken if maximum frame expansion is already required by the rewriter for the TX mirror probe port. In this case enabling remote mirror VLAN tags or encapsulation would add an extra frame expansion, which can cause frame disruption on the mirror destination port. It is not possible to add remote mirror encapsulation to a frame that already has encapsulation added by ES0. In this case the remote mirror encapsulation ID will replace the ID selected by the ES0.ENCAP_ID action.

3.28.14 MPLS Editing

The rewriter performs the following steps related to MPLS editing for each frame and destination.

1. ES0 lookup. ES0 is looked up for each of the frame's destination ports. The action from ES0 controls the MPLS editing.
2. MPLS popping. The existing MPLS link layer header and MPLS label stack is popped.
3. MPLS encapsulation table lookup using the ES0 actions. This table returns the new encapsulation as well as configuration that control MPLS remarking.
4. MPLS pushing. A new MPLS link layer header and MPLS label stack is pushed.
5. MPLS remarking. The VLAN tag headers in the MPLS link layer and MPLS labels are remarked.

3.28.14.1 MPLS ES0 Lookup

The MPLS editing is controlled by the ES0 actions ENCAP_ID and POP_CNT. If there is no ES0 hit, then the rewriter can only do MPLS popping, because MPLS pushing and MPLS remarking require an ES0 hit.

3.28.14.2 MPLS Popping

The rewriter initially pops a number of bytes from the MPLS link layer header and MPLS label stack as specified by either the IFH.DST.ENCAP.W16_POP_CNT value received with the frame from the MPLS classifier or by the ES0 action POP_CNT. The ES0 action takes precedence if non-zero, but it cannot pop more bytes than the pop count received from ANA.

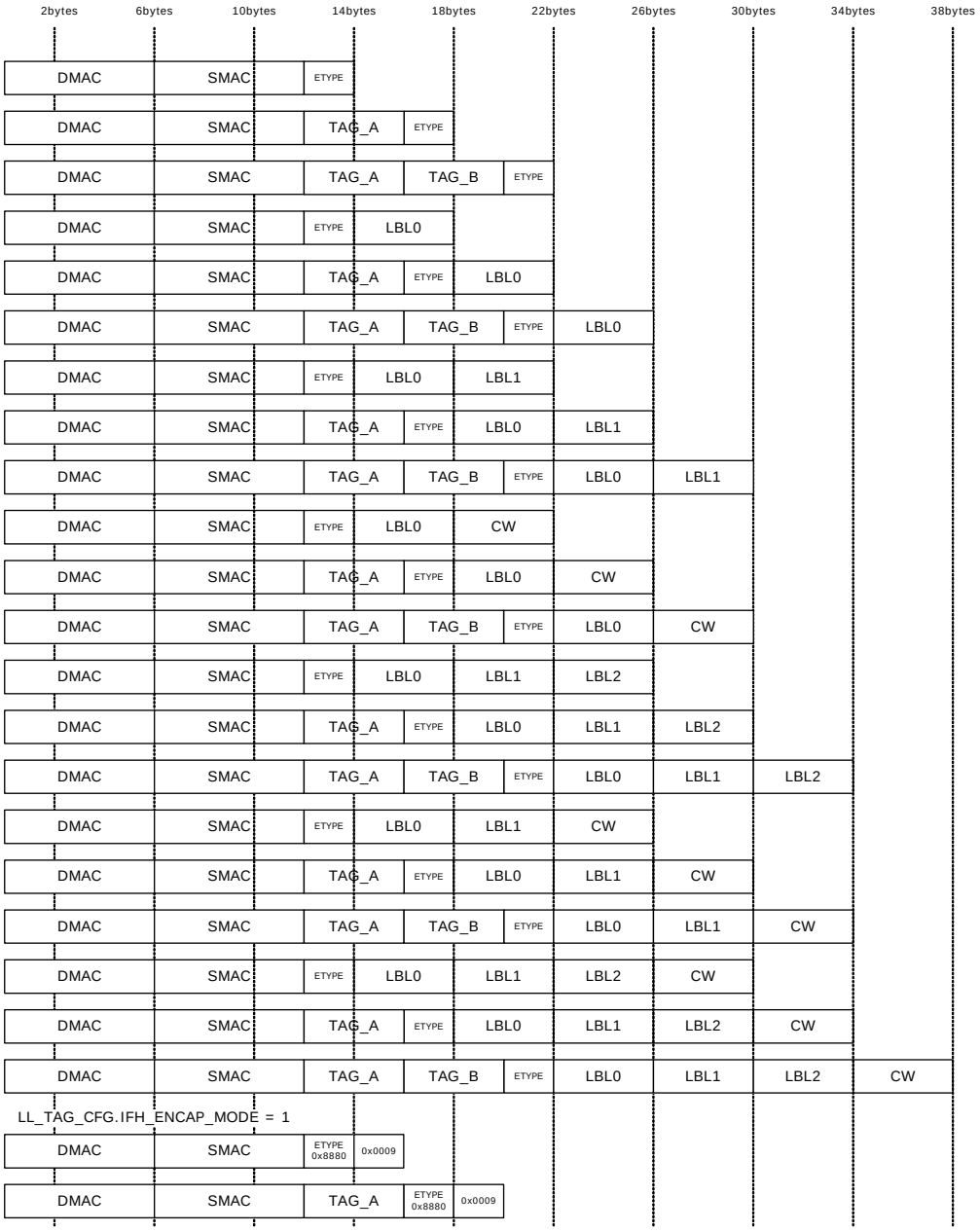
Up to 42 bytes (the MPLS link layer header, three VLAN tags, three MPLS labels, and a control word) can be popped from the frame.

3.28.14.3 MPLS Encapsulation Table

The MPLS encapsulation table is programmed with REW:ENCAP:<FIELD_REG:m> <FIELD>. The index into the MPLS encapsulation table must correspond to the ENCAP_ID used in the associated ES0 action. ENCAP_ID=0 is not available.

The following illustration shows possible encapsulations.

Figure 117 • **MPLS Encapsulation Data**



3.28.14.4 MPLS Encapsulation Configuration

The following table shows the MPLS Encapsulation configuration registers.

Table 189 • **MPLS Encapsulation Configuration Registers**

Register	Description	Replication
LL_DMAC_MSB. DMAC_MSB	16 MSB bits of MPLS link layer DMAC.	128
LL_DMAC_LSB. DMAC_LSB	32 LSB bits of MPLS link layer DMAC.	128
LL_SMAC_MSB.SMAC_MSB	16 MSB bits of MPLS link layer SMAC.	128

Table 189 • **MPLS Encapsulation Configuration Registers *(continued)***

Register	Description	Replication
LL_SMAC_LSB.SMAC_LSB	32 LSB bits of MPLS link layer SMAC.	128
LL_ETYPE.ETYPE	MPLS Link Ethertype.	128
MPLS_LABEL_CFG. INNER_LBL_SEL	Select inner label. Use ES0 to configure the inner most label instead of the encapsulation table.	128
MPLS_LABEL_CFG. LABEL_CNT	Configure the number of MPLS labels to use. Up to three labels are supported.	128
MPLS_LABEL_CFG.CW_ENA	Enable insertion of control word.	128
RSV_LABEL_CFG: RSV_LBL_ENA	Enable reserved MPLS label insertion for MPLS-OAM frames. When this bit is set, an additional MPLS label is inserted if CW insertion is disabled and IFH.DST.PDU_TYPE=OAM_MPLS_TP or IFH.DST.PDU_TYPE=Y1731 and IFH.DST.TYPE_AFTER_POP=CW Note that the reserved label can only be inserted if a CW is not inserted in the frame.	128
RSV_LABEL_CFG: RSV_LBL_POS	Select position of the reserved label. It can be added before or after the last MPLS label 0: Add before last label. 1: Add after last label like the CW.	128
RSV_LABEL_CFG: RSV_TC_SEL	Configures TC-field used in the reserved label. 0: Classified TC. 1: RSV_LABEL_VAL.RSV_TC_VAL. 2-3: Reserved. 4: Mapped using mapping table 0, otherwise use RSV_LABEL_VAL.RSV_TC_VAL. 5: Mapped using mapping table 1, otherwise use mapping table 0. 6: Mapped using mapping table 2, otherwise use RSV_LABEL_VAL.RSV_TC_VAL. 7: Mapped using mapping table 3, otherwise use mapping table 2.	128
CW_VAL.CW_VAL	Control word.	128
LABEL_VAL.LABEL_VAL	Label field value in MPLS label *n*.	128 × 3
LABEL_VAL.TC_VAL	TC value of MPLS label *n*.	128 × 3
LABEL_VAL.SBIT_VAL	SBIT value of MPLS label *n*.	128 × 3
LABEL_VAL.TTL_VAL	TTL value of MPLS label *n*.	128 × 3
RSV_LABEL_VAL. RSV_LBL_VAL	Label field value in the reserved MPLS label.	128
RSV_LABEL_VAL. RSV_TC_VAL	TC value in the reserved MPLS label.	128
RSV_LABEL_VAL. RSV_SBIT_VAL	SBIT value in the reserved MPLS label.	128
RSV_LABEL_VAL. RSV_TTL_VAL	TTL value in the reserved MPLS label.	128
MPLS_REMARK_CFG. LBL_SEL	Used for selecting value of label *n*. 0: LABEL_VAL[*n*].LABEL_VAL. 1-3: Reserved. 4: Mapped using mapping table 0. 5: Mapped using mapping table 1. 6: Mapped using mapping table 2. 7: Mapped using mapping table 3.	128 × 3

Table 189 • **MPLS Encapsulation Configuration Registers *(continued)***

Register	Description	Replication
MPLS_REMARK_CFG. TC_SEL	Used when mapping TC field of MPLS *n*. 0: Classified TC. 1: LABEL_VAL[*n*].TC_VAL. 2. COSID (IFH.VSTAX.MISC.COSID) 3: Reserved 4: Mapped using mapping table 0, otherwise use LABEL_VAL[*n*].TC_VAL. 5: Mapped using mapping table 1, otherwise use mapping table 0. 6: Mapped using mapping table 2, otherwise use LABEL_VAL[*n*].TC_VAL 7: Mapped using mapping table 3, otherwise use mapping table 2.	128 × 3
MPLS_REMARK_CFG. SBIT_SEL	Configures the S-bit used in label *n*. 0: Classified SBIT (IFH.DST.ENCAP.MPLS_SBIT). 1: Fixed: LABEL_VAL[*n*].SBIT_VAL. 2: Mixed: Use Classified SBIT if IFH.DST.ENCAP.PDU_TYPE = OAM_MPLS_TP else LABEL_VAL[*n*].SBIT_VAL. 3: Reserved.	128 × 3
MPLS_REMARK_CFG. TTL_SEL	Configure the TTL-field used in label *n*. 0: Classified TTL (IFH.DST.ENCAP.MPLS_TTL). 1: Fixed: LABEL_VAL[*n*].TTL_VAL. 2: Mixed: Use Classified TTL if IFH.DST.ENCAP.PDU_TYPE = OAM_MPLS_TP else LABEL_VAL[*n*].TTL_VAL. 3: Reserved.	128 × 3
LL_TAG_CFG. IFH_ENCAP_MODE	Enables IFH encapsulation mode for this entry. The frame link layer format is changed to [LL_DMAC][LL_SMAC][0x8880][0x0009]. Optionally, one VLAN tag can be added if LL_TAG_CFG.TAG_CFG = 1 [LL_DMAC][LL_SMAC][LL_TAG:A][0x8880][0x000 9] No other encapsulation fields are used in this mode.	128
LL_TAG_CFG. TAG_CFG	Selects the number of VLAN tags the MPLS encapsulation data contains after the SMAC. Up to two tags are supported.	128
LL_TAG_VAL. TAG_VID_VAL	Configures VID of encapsulation tags. Idx0: TAG_A. Idx1: TAG_B.	128 × 2
LL_TAG_VAL. TAG_DEI_VAL	Configures DEI of encapsulation tags. Idx0: TAG_A. Idx1: TAG_B.	128 × 2
LL_TAG_VAL. TAG_PCP_VAL	Configures PCP of encapsulation tags. Idx0: TAG_A. Idx1: TAG_B.	128 × 2
LL_TAG_VAL. TAG_TPID_VAL	Configures TPID of encapsulation tags. Idx0: TAG_A.Idx1: TAG_B.	128 × 2

Table 189 • **MPLS Encapsulation Configuration Registers *(continued)***

Register	Description	Replication
LL_TAG_REMARK_CFG. TAG_DEI_SEL	Selects DEI for MPLS encapsulation tag. Idx0: TAG_A. Idx1: TAG_B. 0: Classified DEI. 1: Encapsulation LL_TAG_VAL[*n*].TAG_DEI_VAL. 2: REW::DP_MAP.DP[IFH.VSTAX.QOS.DP]. 3: Reserved. 4: Mapped using mapping table 0, otherwise use LL_TAG_VAL[*n*].TAG_DEI_VAL. 5: Mapped using mapping table 1, otherwise use mapping table 0. 6: Mapped using mapping table 2, otherwise use LL_TAG_VAL[*n*].TAG_DEI_VAL. 7: Mapped using mapping table 3, otherwise use mapping table 2.	128 × 2
LL_TAG_REMARK_CFG. TAG_PCP_SEL	Selects PCP for MPLS encapsulation tag. Idx0: TAG_A Idx1: TAG_B. 0: Classified PCP. 1: Encapsulation LL_TAG_VAL[*n*].TAG_PCP_VAL. 2-3: Reserved. 4: Mapped using mapping table 0, otherwise use LL_TAG_VAL[*n*].TAG_PCP_VAL. 5: Mapped using mapping table 1, otherwise use mapping table 0. 6: Mapped using mapping table 2, otherwise use LL_TAG_VAL[*n*].TAG_PCP_VAL. 7: Mapped using mapping table 3, otherwise use mapping table 2.	128 × 2
LL_TAG_REMARK_CFG. TAG_VID_SEL	Selects VID of MPLS encapsulation tag. Idx0: TAG_A Idx1: TAG_B. 0: Classified VID + LL_TAG_VAL[*n*].TAG_VID_VAL. 1: LL_TAG_VAL[*n*].TAG_VID_VAL.	
LL_TAG_REMARK_CFG. TAG_TPID_SEL	Selects TPID for MPLS encapsulation tag. Idx0: TAG_A Idx1: TAG_B. 0: Encapsulation LL_TAG_VAL[*n*].TAG_TPID. 1: Classified. ANA controls using IFH: If IFH.DST.ENCAP.TAG_TPID = STD_TPID: If IFH.VSTAX.TAG_TYPE = 0, then 0x8100 else LL_TAG_VAL[*n*].TAG_TPID. If IFH.DST.ENCAP.TAG_TPID = CUSTOM: Custom TPID 1 to 3 are configured by REW::TPID_CFG[*n*].TPID_VAL.	128 × 2

3.28.14.5 MPLS Pushing

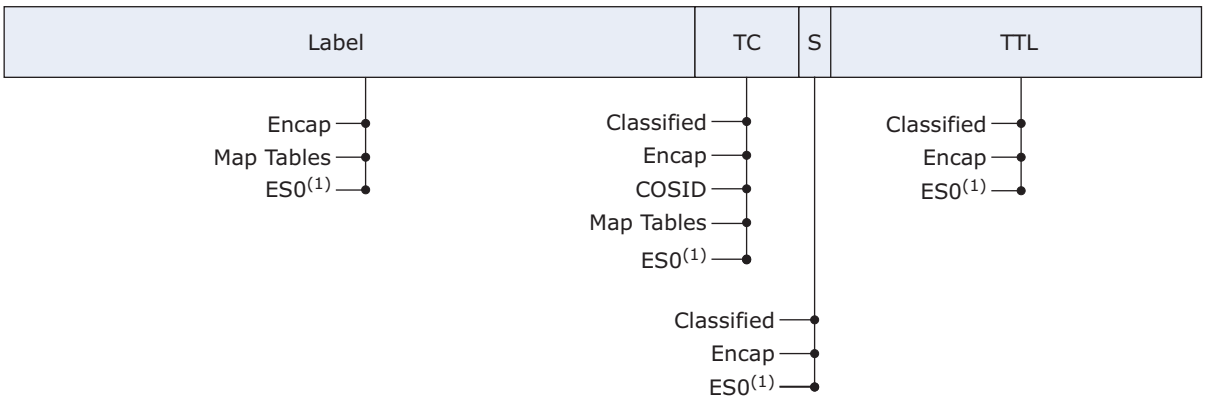
The ES0 action ENCAP_ID enables MPLS pushing. If ENCAP_ID = 0, MPLS pushing is effectively disabled.

3.28.14.6 MPLS Remarking

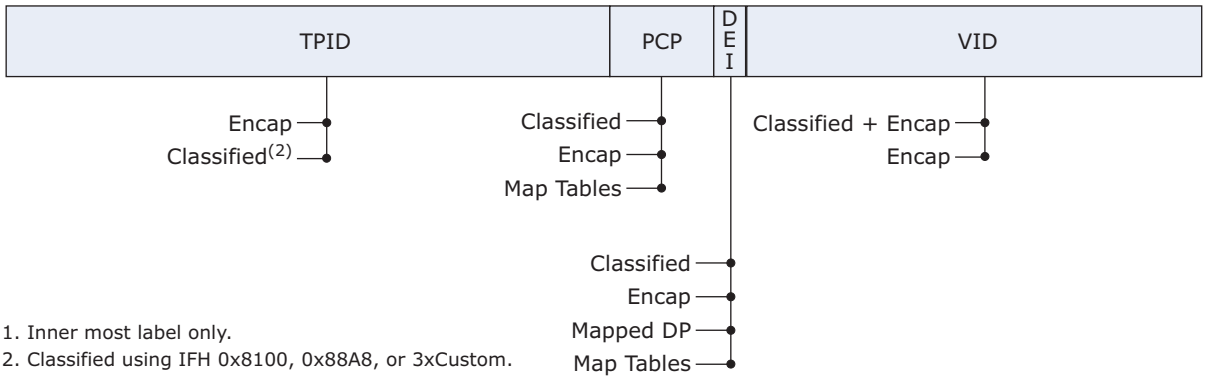
The rewriter can remark certain parameters in the pushed MPLS data. The following illustration shows an overview of the available remark options.

***Figure 118* • MPLS Remarking**

Up to three remark options for each MPLS label:



Remark options for MPLS Link Layer VLAN Tag:



1. Inner most label only.
2. Classified using IFH 0x8100, 0x88A8, or 3xCustom.

3.28.14.6.1 MPLS Link Layer Tag Remarking

The link layer VLAN tags are remarked similarly to the Ethernet A to C tags.

3.28.14.6.2 MPLS Label Remark Options

The MPLS labels are formatted based on ES0 actions and the following configurations for each ENCAP_ID.

- The inner most label can be controlled directly from ES0 actions if enabled by MPLS_LABEL_CFG.INNER_LBL_SEL. ES0 has actions that are similar to the normal MPLS remark options for control of all label fields.
- Control word disable. ES0 can directly disable insertions of the CW regardless of MPLS_LABEL_CFG.CW_ENA configuration.
- The Label value can be obtained from the mapping tables or directly from the ENCAP table depending on the MPLS_REMARK_CFG.LBL_SEL configuration.
- Add a reserved MPLS label instead of the control word for OAM MPLS-TP frames. The reserved label can be added before or after the last MPLS label in the label stack.
- Remark the S-bit in any of the up to three MPLS labels with the frame's classified S-bit from the MPLS classifier. Each label is controlled independently of the others. If MPLS_REMARK[*n*].SBIT_SEL is 0 for one of the MPLS labels, the S-bit received from the MPLS classifier with the frame overwrites the S-bit in the pushed MPLS label.
- Remark the TTL value in any of the up to three MPLS labels with the frame's classified TTL value from the MPLS classifier. Each label is controlled independently of the others. If MPLS_REMARK[*n*].TTL_SEL is 0 for one of the MPLS labels, the TTL value received from the MPLS classifier together with the frame overwrites the TTL value in the pushed MPLS label.
- Remark the TC bits in any of the up to three MPLS labels based on the generic mapping functions controlled by ES0. Each label is controlled independently of the others.

- Remark the TC bits in any of the up to three MPLS labels with the frame's classified TC bits from the MPLS classifier. Each label is controlled independently of the others. If TC_SEL[*n*] is 0, the TC bits received from the MPLS classifier together with the frame overwrites the TC bits in the pushed MPLS label.

3.28.15 Internal Frame Header Insertion

The rewriter can be configured to insert the internal frame header (IFH) into the frame upon transmission. For more information about the IFH, Frame Headers, page 22.

Insertion of the IFH can be configured on any port, but it is intended for frames forwarded to the internal CPU and the NPI port (external CPU). When IFH insertion is enabled on a given port, all frames leaving that port contain the IFH.

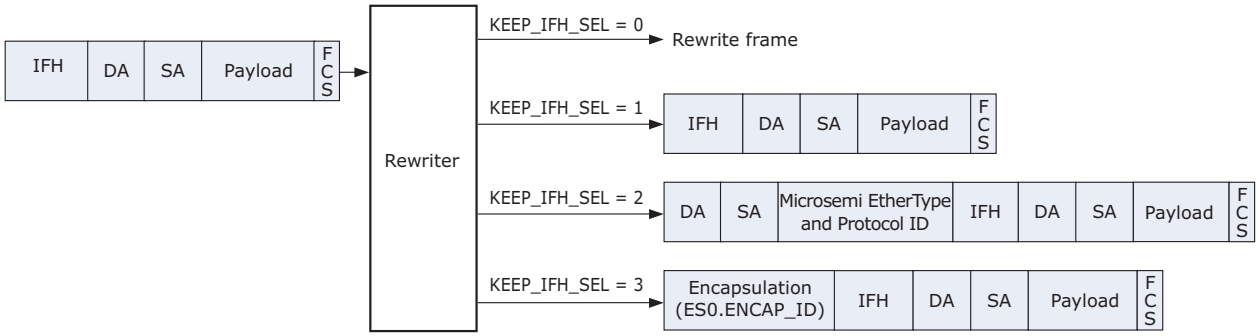
The following table lists the registers for configuring the insertion of extraction IFH into the frame.

Table 190 • **IFH Configuration Registers**

Register	Description	Replication
PORT_CTRL.KEEP_IFH_SEL	Configures the rewriter IFH mode for the port. See also ASM::PORT_CFG.INJ_FORMAT_CFG. 0: Normal mode. 1: Keep IFH without modifications. Frames are not updated. IFH is kept. 2: Encapsulate IFH. The frame's DMAC, SMAC and a fixed TAG with ETYPE = 8880 (Microsemi) and EPID=0x0009 are inserted in front of the IFH: [FRM_DMAC][FRM_SMAC][0x8880][0x0009][IFH][FRAME]. 3: Encapsulate IFH using the ENCAP table. Use ES0 to generate an ENCAP_ID and insert the encapsulation in front of the IFH:[ENCAP][IFH][FRAME].	Ports

The following illustration shows supported formats using KEEP_IFH_SEL.

Figure 119 • **Supported KEEP_IFH_SEL Formats**



Frames are always formatted using KEEP_IFH_SEL = 1 when extracting frames to the internal CPU ports. This cannot be changed.

Note that for KEEP_IFH_SEL = 1, the frames transmitted on external ports are not valid Ethernet frames, because the extraction IFH precedes the DMAC. As a result, this frame format must be used for direct transmission to another VSC7430 device that understands the format or to a dedicated FPGA or NPU. This format cannot be used for forwarding to a standard Ethernet device.

The frame formats using KEEP_IFH_SEL = 2 or KEEP_IFH_SEL = 3 are primarily used on a port that is directly attached to an external CPU. When the IFH is included using these frame formats, the IFH is included in the frame's FCS calculation and a valid Ethernet frame is transmitted.

3.28.16 Frame Injection from Internal CPU

To instruct the switch core how to process the frame, the internal CPU injects frames to switch core using the internal frame header. The format of injected frames from the internal CPU is similar to the KEEP_IFH_SEL = 1 format.

On arrival to the rewriter, such injected frames are no different from any other frames. However, the IFH contains information related to rewriter functionality specifically targeting efficient frame injection:

- Do not rewrite (IFH.FWD.DO_NOT_REW):
  If this injection IFH bit is set, the rewriter forward frames unchanged to the egress port no matter any other rewriter configurations, except IFH.FWD.update_fcs.
- Update frame check sequence (IFH.FWD.UPDATE_FCS):
  If this injection IFH bit is set, the rewriter updates the FCS before forwarding a frame to the egress port. By setting this bit in the Injection IFH, the internal CPU can leave it to switch core hardware to make sure the FCS value included with the frame is correct even though no other frame modifications are performed by the switch core.
- Swap MACs (IFH.DST.ENCAP.SWAP_MAC):
  Swap MACs in the Ethernet link layer and clear bit 40 of the new SMAC before sending the frame to the egress port. Swapping the MACs will also force the frame FCS to be updated. Cannot be used if IFH.FWD.DO_NOT_REW = 1.

3.29 Disassembler

This section provides information about the disassembler (DSM) block. The disassembler is mainly responsible for disassembling cells into Taxi words and forwarding them to the port modules. In addition, it has several other responsibilities such as:

- MAC Control sublayer PAUSE function with generation of PAUSE frames and stop forwarding traffic on reception of PAUSE frames
- Aging of frames that are stopped by flow control
- Controlling transmit data rate according to IEEE802.3ar

The following table lists replication terminology.

Table 191 • **Replication Terminology**

Replication Terminology	Description	Replication Value
Per port	Per physical ports. For example, DEV1Gs, DEV2G5s, DEV10Gs and CPU ports.	15

3.29.1 Setting Up Ports

In general, no additional configuration is required to bring up a port in the disassembler. However, the following are some exceptions:

- For the 10G capable ports, set DSM::BUF_CFG.CSC_STAT_DIS to 1 when the port is set up for 10 Gbps, because the collection of statistics are handled locally in the DEV10G. For lower speeds, the port uses a DEV2G5.
- For 10G capable ports, set DSM::DEV_TX_STOP_WM_CFG.DEV_TX_STOP_WM to 3 if the port is set up for 2.5 Gbps or 1 Gbps, and set it to 1 for 100 Mbps and 10 Mbps.
- For 10G capable ports, set DSM::DEV_TX_STOP_WM_CFG.DEV10G_SHADOW_ENA to 1 when port is set up for speeds below 10 Gbps.
- For 2.5G capable ports, the DSM::DEV_TX_STOP_WM_CFG.DEV_TX_STOP_WM must be set to 1 if the port is set up for 100 Mbps and 10 Mbps.

Ports are indexed by 0 – 10 being front ports and 11 – 12 being CPU ports.

The following table lists the registers associated with setting up basic ports.

Table 192 • **Basic Port Setup Configuration Registers Overview**

Target::Register.Field	Description	Replication
DSM::BUF_CFG.CSC_STAT_DIS	Disables collection of statistics in the disassembler.	Per port
DSM::DEV_TX_STOP_WM_CFG.DEV_TX_STOP_WM	Watermark for maximum fill level of port module TX buffer.	Per port
DSM::DEV_TX_STOP_WM_CFG.DEV10G_SHADOW_ENA	Enable low speeds for 10G capable ports.	Per port

3.29.2 Maintaining the Cell Buffer

The cell buffer is responsible for aligning and buffering the cell data received from the rewriter before the data are forwarded on the Taxi bus.

The cell buffer can be flushed for each port separately.

The following table lists registers associated with configuring buffer maintenance.

Table 193 • **Buffer Maintenance Configuration Register Overview**

Target::Register.Field	Description	Replication
DSM::CLR_BUF.CLR_BUF	Flush the cell buffer.	Per port

Before a buffer is flushed, ensure that no data is forwarded to this port by stopping the QSYS forwarding and disabling FC/PFC so that frames waiting for transmission will be sent. FC can be disabled in DSM::RX_PAUSE_CFG.RX_PAUSE_EN.

3.29.3 Setting Up MAC Control Sublayer PAUSE Function

This section provides information about setting up the MAC control sublayer PAUSE function.

3.29.3.1 PAUSE Frame Generation

The main sources for triggering generation of PAUSE frames are the port level and buffer level watermarks in the queue system. It is also possible to trigger PAUSE generation based on the analyzer policing state.

For watermark-based PAUSE generation, a hysteresis is implemented. FC is indicated when the level of used resources exceeds the high watermark and it is released when the level falls below the low watermark. The following table lists the registers associated with configuration PAUSE frame generation.

Table 194 • **PAUSE Frame Generation Configuration Registers Overview**

Target::Register.Field	Description	Replication
DSM::ETH_FC_CFG.FC_ANA_ENA	Controls generation of FC based on FC request from the analyzer.	Per port
DSM::ETH_FC_CFG.FC_QS_ENA	Controls the generation of FC based on FC request from the queue system.	Per port
DSM::MAC_CFG.TX_PAUSE_VAL	The pause_time as defined by IEEE802.3 Annex 31B2.	Per port
DSM::MAC_CFG.TX_PAUSE_XON_XOFF	TX PAUSE zero on deassert.	Per port
DSM::MAC_ADDR_BASE_HIGH_CFG.MAC_ADDR_HIGH	Bits 47-24 of SMAC address used in MAC_Control frames.	Per port
DSM::MAC_ADDR_BASE_LOW_CFG.MAC_ADDR_LOW	Bits 23-0 of SMAC address used in MAC_Control frames.	Per port

DSM::MAC_CFG.TX_PAUSE_VAL defines the timer value for generated PAUSE frames. If the state does not change by half of the time specified in this bit group, a new PAUSE frame is generated.

To generate a zero value pause frame when the reason for pausing has disappeared, set DSM::MAC_CFG.TX_PAUSE_XON_XOFF to 1.

The DMAC of a generated pause frame is always the globally assigned 48-bit multicast address 01 80 C2 00 00 01.

The SMAC of a generated pause frame is defined by DSM::MAC_ADDR_BASE_HIGH_CFG.MAC_ADDR_HIGH and DSM::MAC_ADDR_BASE_LOW_CFG.MAC_ADDR_LOW.

3.29.3.2 Reaction on Received PAUSE Frame

The assembler detects received PAUSE frames and forwards the pause_time to the disassembler, which stops data transmission (if enabled) for the period defined by pause_time. If PFC is enabled for a port, then RX_PAUSE_EN must be disabled in the disassembler.

The following table lists the registers associated with configuring PAUSE frame reception.

Table 195 • **PAUSE Frame Reception Configuration Registers Overview**

Target::Register.Field	Description	Replication
DSM::RX_PAUSE_CFG.RX_PAUSE_EN	Enables flow control in Rx direction	Per port

3.29.3.3 PFC Pause Frame Generation

When PFC is enabled for a port, PFC PAUSE frames are generated when requested by the queue system.

PFC is enabled on a per port basis in the disassembler. The queue system must also be setup to generate PFC requests towards the disassembler. Regular flow control and PFC cannot operate simultaneously on the same port. The available PFC configurations in the disassembler are listed in the following table.

Table 196 • **PFC PAUSE Frame Generation Configuration Registers Overview**

Target::Register.Field	Description	Replication
DSM::ETH_PFC_CFG.PFC_MIN_UPDATE_TIME	Minimum time between two PFC PDUs when PFC state changes after transmission of PFC PDU.	Per port
DSM::ETH_PFC_CFG.PFC_XOFF_MIN_UPDATE_ENA	After sending PFC PDU with flow control deasserted for all priorities, enforces a PFC_MIN_UPDATE_TIME delay before allowing transmission of next PFC PDU.	Per port
DSM::ETH_PFC_CFG.PFC_ENA	Enables PFC operation for the port.	Per port

3.29.4 Setting up Flow Control in Half-Duplex Mode

The following table lists the configuration registers for half-duplex mode flow control. For more information about setting up flow control in half-duplex mode, see PAUSE Frame Generation, page 360.

Table 197 • **Half-Duplex Mode Flow Control Configuration Register Overview**

Target::Register.Field	Description	Replication
DSM::MAC_CFG.HDX_BACKPRESSURE	Enables HDX backpressure instead of FDX FC when FC is generated.	Per port

To enable half-duplex flow control, DSM::MAC_CFG.HDX_BACKPRESSURE must be set to 1. In this mode, the disassembler forwards the FC status to the port modules, which then collides incoming frames.

3.29.5 Setting Up Frame Aging

The following tables provide the aging configuration and status register settings. If frame aging is enabled, the disassembler discards frames that are stuck, for example, due to flow control or frames that are received from the queue system with an era value eligible for aging. Frames discarded by aging are counted in DSM::AGED_FRMS.AGED_FRMS_CNT.

Table 198 • **Aging Configuration Register Overview**

Target::Register.Field	Description	Replication
DSM::BUF_CFG.AGING_ENA	Enable aging of frames stuck in the disassembler buffer system for long periods.	Per port

Table 199 • **Aging Status Register Overview**

Target::Register.Field	Description	Replication
DSM::AGED_FRMS.AGED_FRMS_CNT	Count number of aged frames.	Per port

3.29.6 Setting Up Transmit Data Rate Limiting

The disassembler can limit the transmit data rate as specified in IEEE802.3ar, which defines the following three rate limiting methods. The disassembler allows enabling of any combination of the following three methods.

- Frame overhead
- Payload data rate
- Frame rate

The following table lists the registers associated with rating limiting.

Table 200 • **Rate Limiting Common Configuration Registers Overview**

Target::Register.Field	Description	Replication
DSM::TX_RATE_LIMIT_MODE. TX_RATE_LIMIT_ACCUM_MODE_ENA	Enables for accumulated rate limit mode.	Per port
DSM::TX_RATE_LIMIT_MODE. PAYLOAD_PREAM_CFG	Defines whether the preamble is counted as payload in txRateLimitPayloadRate and txRateLimitFrameRate mode.	Per port
DSM::TX_RATE_LIMIT_MODE. PAYLOAD_CFG	Defines if what is configured as header size in TX_RATE_LIMIT_HDR_SIZE:: TX_RATE_LIMIT_HDR_CFG is subtracted form the payload.	Per port
DSM::TX_RATE_LIMIT_MODE. IPG_SCALE_VAL	Scales the IPG calculated by txRateLimitFrameOverhead and/or txRateLimitPayloadRate by a power of 2.	Per port
DSM::TX_RATE_LIMIT_HDR_CFG. TX_RATE_LIMIT_HDR_SIZE	Defines how much of the frame is seen as header and not counted as payload.	per port

If more than one of the rate limiting modes previously mentioned are enabled, the additional inter-packet gap is the maximum of each of the values generated by one of the enabled modes. However, if only the frame overhead and the data payload data rate modes are enabled, the additional inter-packet gap can be calculated as the sum of the additional gaps retrieved from each of the two modes. To enable the function, set DSM::TX_RATE_LIMIT_MODE.TX_RATE_LIMIT_ACCUM_MODE_ENA to 1.

If the preamble of a frame is not to be counted as payload, set DSM::TX_RATE_LIMIT_MODE.PAYLOAD_CFG to 0.

In addition, if parts of the frame are to be seen as header and not counted as payload, set DSM::TX_RATE_LIMIT_MODE.PAYLOAD_CFG to 1.
DSM::TX_RATE_LIMIT_HDR_CFG.TX_RATE_LIMIT_HDR_SIZE defines how much of the frame that

should be considered as header. Note that this register is global for all ports enabled by DSM::TX_RATE_LIMIT_MODE.PAYLOAD_CFG. The payload configuration applies to Payload Data Rate and Frame Rate mode.

To enable bandwidth limiting down to very low data rates, the calculated IPG can be scaled. In other words, multiplied by a power of 2 in the range 2 to 1024. By this it is possible to limit the data rate down to 2.3% (for 1536 byte frames).

3.29.6.1 Setting Up Frame Overhead

The following table lists the frame overhead configuration registers.

Table 201 • **Rate Limiting Frame Overhead Configuration Registers Overview**

Target::Register.Field	Description	Replication
DSM::TX_RATE_LIMIT_MODE. TX_RATE_LIMIT_FRAME_OVERHEAD_ENA	Enable txRateLimitFrameOverhead mode.	Per port
DSM::RATE_CTRL.FRM_GAP_COMP	Inter-packet gap to be used.	Per port

To increase the minimum inter-packet gap for all packets, set DSM::TX_RATE_LIMIT_MODE.TX_RATE_LIMIT_FRAME_OVERHEAD_ENA to 1, and set DSM::RATE_CTRL.FRM_GAP_COMP to a value between 13 to 255.

For more information about the scaling feature if the GAP must be above 255, see Setting Up Transmit Data Rate Limiting, page 362.

3.29.6.2 Setting Up Payload Data Rate

The following table lists the payload data rate configuration registers.

Table 202 • **Rate Limiting Payload Data Rate Configuration Registers Overview**

Target::Register.Field	Description	Replication
DSM::TX_RATE_LIMIT_MODE. TX_RATE_LIMIT_PAYLOAD_RATE_ENA	Enable txRateLimitPayloadRate mode.	Per port
DSM::TX_IPG_STRETCH_RATIO_CFG. TX_FINE_IPG_STRETCH_RATIO	Stretch ratio.	Per port

To limit the data rate relative to the transmitted payload, set DSM::TX_RATE_LIMIT_MODE.TX_RATE_LIMIT_PAYLOAD_RATE_ENA to 1.

The inter-packet gap increase can be expressed as:

$\Delta IPG = 2^S \times (256/R) \times L$

Where:

S: Scaling factor DSM::TX_RATE_LIMIT_MODE.IPG_SCALE_VAL

R: Stretch ratio DSM::TX_IPG_STRETCH_RATIO_CFG.TX_FINE_IPG_STRETCH_RATIO

L: Payload length, possibly modified according to DSM::TX_RATE_LIMIT_MODE.PAYLOAD_PREAM_CFG and DSM::TX_RATE_LIMIT_MODE.PAYLOAD_CFG

Notes

Total IPG is 12 byte standard IPG plus IPG.

Values for R must not be below 1152 or above 518143.

Values for S must not be above 10.

Fractional parts of the resulting IPG value are carried forward to the next IPG.

Higher values for S allow for finer target accuracy/granularity on the cost of an increased IPG jitter.

The utilization can be expressed as:

U = L / (L + IPG) or

U = R / (R + 2S × 256)

3.29.6.3 Setting Up Frame Rate

The following table lists the frame rate configuration registers.

Table 203 • **Rate Limiting Frame Rate Registers Overview**

Target::Register.Field	Description	Replication
DSM::TX_RATE_LIMIT_MODE. TX_RATE_LIMIT_FRAME_RATE_ENA	Enables txRateLimitFrameRate mode.	Per port
DSM::TX_FRAME_RATE_START_CFG. TX_FRAME_RATE_START	Frame rate start.	Per port

To define a minimum frame spacing, set DSM::TX_RATE_LIMIT_MODE.TX_RATE_LIMIT_FRAME_RATE_ENA to 1.

At start of payload, a counter is loaded with the value stored in DSM::TX_FRAME_RATE_START_CFG.TX_FRAME_RATE_START.

What is counted as payload is defined by DSM::TX_RATE_LIMIT_MODE.PAYLOAD_PREAM_CFG and DSM::TX_RATE_LIMIT_MODE.PAYLOAD_CFG.

With every payload byte transmitted, the counter is decremented by 1.

At end of frame, the GAP will be extended by the counter value, if above 12.

3.29.7 Error Detection

The following table lists the error detection status registers.

Table 204 • **Error Detection Status Registers Overview**

Target::Register.Field	Description	Replication
DSM::BUF_OFLW_STICKY.BUF_OFLW_STICKY	Cell buffer had an overflow.	Per port
DSM::BUF_UFLW_STICKY.BUF_UFLW_STICKY	Cell buffer had an underrun.	Per port
DSM::TX_RATE_LIMIT_STICKY.TX_ RATE_LIMIT_STICKY	IPG was increased by one of the three rate limiting modes.	Per port.

If one or both of BUF_OFLW_STICKY and BUF_UFLW_STICKY sticky bits are set, the port module was set up erroneously.

TX_RATE_LIMIT_STICKY is set when an IPG of a frame is increased due to one of the three rate limiting modes.

3.30 Layer 1 Timing

There are eight recovered clocks, four outputs that provide timing sources for external timing circuitry in redundant timing implementations, and four internal clocks for the timing-recovery circuit. The following tables list the registers and pins associated with Layer 1 timing.

Table 205 • **Layer 1 Timing Configuration Registers**

Register	Description
HSIO::SYNC_ETH_CFG	Configures recovered clocks. Replicated per recovered clock.

Table 205 • **Layer 1 Timing Configuration Registers *(continued)***

Register	Description
HSIO::SYNC_ETH_PLL_CFG	Additional PLL recovered clock configuration. Replicated per recovered clock.
HSIO::PLL5G_CFG3	Enables high speed clock output.
PHY[0]:PHY_GP:PHY_RCVD_CLK0_CTRL	Configures PHY0 recovered clock.
PHY[1]:PHY_GP:PHY_RCVD_CLK1_CTRL	Configuration PHY1 recovered clock.

Table 206 • **Layer 1 Timing Recovered Clock Pins**

Pin Name	I/O	Description
RCVRD_CLK0	O	Recovered clock output, configured by SYNC_ETH_CFG[0]. This is an overlaid function on GPIO.
RCVRD_CLK1	O	Recovered clock output, configured by SYNC_ETH_CFG[1]. This is an overlaid function on GPIO.
RCVRD_CLK2	O	Recovered clock output, configured by SYNC_ETH_CFG[2]. This is an overlaid function on GPIO.
RCVRD_CLK3	O	Recovered clock output, configured by SYNC_ETH_CFG[3]. This is an overlaid function on GPIO.
CLKOUTPLL	O	PLL high speed clock output.
CLKOUTPLL2	O	PLL2 high speed clock output.

It is possible to recover receive timing from any 10/100/1000 Mbps, or 2.5 Gbps data streams into the device.

The recovered clock outputs have individual divider configuration (through SYNC_ETH_CFG.SEL_RCVRD_CLK_DIV) to allow division of SerDes receive frequency by 1, 2, 4, 5, 8, 16, or 25.

The recovered clocks are single-ended outputs, and the suggested divider settings in the following tables are selected to make sure to not output too high a frequency via the input and outputs.

Ports operating in 1 Gbps or lower speeds allow recovery of either 31.25 MHz or 125 MHz clocks. Configure the ports as shown in the following table.

Table 207 • **Recovered Clock Settings for 1 Gbps or Lower**

Port	Output Frequency	Register Settings for Output *n*
CuPHY 0-1	125 MHz	Only for internal clocks (*n* = 4, 5, 6, and 7). SYNC_ETH_CFG[*n*].RCVRD_CLK_ENA=1, SYNC_ETH_CFG[*n*].SEL_RCVRD_CLK_DIV=6, SYNC_ETH_CFG[*n*].SEL_RCVRD_CLK_SRC=(DEV), PHY_RCVD_CLK[DEV]_CTRL.RCVD_CLK[DEV]_ENA=1, PHY_RCVD_CLK[DEV]_CTRL.CLK_SRC_SEL[DEV]=[DEV], PHY_RCVD_CLK[DEV]_CTRL.CLK_FREQ_SEL[DEV]=1, and PHY_RCVD_CLK[DEV]_CTRL.CLK_SEL_PHY[DEV]=1

Table 207 • **Recovered Clock Settings for 1 Gbps or Lower**

Port	Output Frequency	Register Settings for Output *n*
CuPHY 0-1	31.25 MHz	SYNC_ETH_CFG[*n*].RCVRD_CLK_ENA=1, SYNC_ETH_CFG[*n*].SEL_RCVRD_CLK_DIV=1, SYNC_ETH_CFG[*n*].SEL_RCVRD_CLK_SRC=(DEV), PHY_RCVD_CLK[DEV]_CTRL.RCVD_CLK[DEV]_ENA=1, PHY_RCVD_CLK[DEV]_CTRL.CLK_SRC_SEL[DEV]=[DEV], PHY_RCVD_CLK[DEV]_CTRL.CLK_FREQ_SEL[DEV]=1, and PHY_RCVD_CLK[DEV]_CTRL.CLK_SEL_PHY[DEV]=1
SerDes 0-6	125 MHz	Only for internal clocks (*n* = 4, 5, 6, and 7). SYNC_ETH_CFG[*n*].RCVRD_CLK_ENA=1, SYNC_ETH_CFG[*n*].SEL_RCVRD_CLK_DIV=6, and SYNC_ETH_CFG[*n*].SEL_RCVRD_CLK_SRC=(DEV+4)
SerDes 0-6	31.25 MHz	SYNC_ETH_CFG[*n*].RCVRD_CLK_ENA=1, SYNC_ETH_CFG[*n*].SEL_RCVRD_CLK_DIV=1, and SYNC_ETH_CFG[*n*].SEL_RCVRD_CLK_SRC=(DEV+4)

Ports operating in 2.5 Gbps mode allow recovery of a 31.25 MHz clock. The following table shows the configuration settings.

Table 208 • **Recovered Clock Settings for 2.5 Gbps**

Port	Output Frequency	Register Settings for Output *n*
5-6	125 MHz	Only for internal clocks (*n* = 4, 5, 6, or 7). SYNC_ETH_CFG[*n*].RCVRD_CLK_ENA=1, SYNC_ETH_CFG[*n*].SEL_RCVRD_CLK_DIV=6, and SYNC_ETH_CFG[*n*].SEL_RCVRD_CLK_SRC=(DEV+4)
5-6	31.25 MHz	SYNC_ETH_CFG[*n*].RCVRD_CLK_ENA=1, SYNC_ETH_CFG[*n*].SEL_RCVRD_CLK_DIV=1, and SYNC_ETH_CFG[*n*].SEL_RCVRD_CLK_SRC=(DEV+4)

The frequency of the PLLs can be used as recovered clock. The following table shows the configurations.

Table 209 • **Recovered Clock Settings for PLL**

PLL	Output Frequency	Register Settings for Output *n*
PLL	125 MHz	Only for internal clocks (*n* = 4, 5, 6, or 7). SYNC_ETH_CFG[*n*].RCVRD_CLK_ENA=1, SYNC_ETH_CFG[*n*].SEL_RCVRD_CLK_DIV=6, and SYNC_ETH_CFG[*n*].SEL_RCVRD_CLK_SRC=12
PLL	31.25 MHz	SYNC_ETH_CFG[*n*].RCVRD_CLK_ENA=1, SYNC_ETH_CFG[*n*].SEL_RCVRD_CLK_DIV=1, and SYNC_ETH_CFG[*n*].SEL_RCVRD_CLK_SRC=12
PLL2	125 MHz	Only for internal clocks (*n* = 4, 5, 6, or 7). SYNC_ETH_CFG[*n*].RCVRD_CLK_ENA=1, SYNC_ETH_CFG[*n*].SEL_RCVRD_CLK_DIV=6, and SYNC_ETH_CFG[*n*].SEL_RCVRD_CLK_SRC=21
PLL2	31.25 MHz	SYNC_ETH_CFG[*n*].RCVRD_CLK_ENA=1, SYNC_ETH_CFG[*n*].SEL_RCVRD_CLK_DIV=1, and SYNC_ETH_CFG[*n*].SEL_RCVRD_CLK_SRC=21

The recovered clock from PLLs can also be sent directly out on the differential high-speed CLKOUTPLL and CLKOUTPLL2 outputs. The recovered clock frequency must be set to copy the switch core using HSIO::PLL5G_CFG3.CLKOUT2_SEL.

It is possible to automatically squelch the clock output when the device detects a loss of signal on an incoming data stream. This can be used for failover in external timing recovery solutions.

The following table lists how to configure squelch for possible recovered clock sources (configured in SYNC_ETH_CFG[*n*].SEL_RCVRD_CLK_SRC).

Table 210 • **Squelch Configuration for Sources**

SRC	Associated Squelch Configuration
4-8	Set SERDES1G_COMMON_CFG.SE_AUTO_SQUELCH_ENA in SD macro to enable squelch when receive signal is lost. SD1G macro index is (SRC-4).
9-10	Set SERDES6G_COMMON_CFG.SE_AUTO_SQUELCH_ENA in SD macro to enable squelch when receive signal is lost. SD6G macro-index is (SRC-9).
12	Set SYNC_ETH_PLL_CFG[0].PLL_AUTO_SQUELCH_ENA to enable squelch when PLL loses lock.
21	Set SYNC_ETH_PLL_CFG[1].PLL_AUTO_SQUELCH_ENA to enable squelch when PLL2 loses lock.

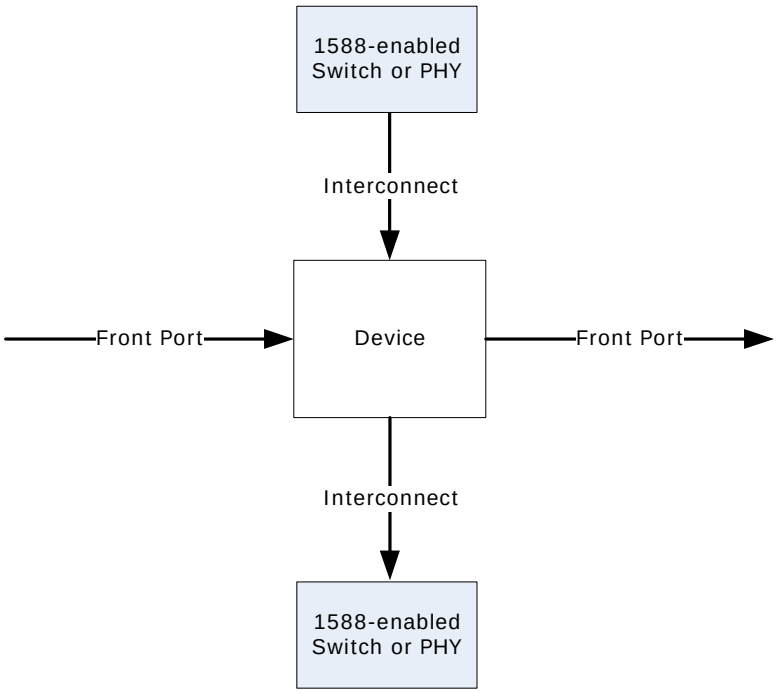
When squelching, the clock will stop when detecting loss of signal (or PLL lock). The clock will stop on either on high or low level.

3.31 Hardware Time Stamping

Hardware time stamping provides nanosecond-accurate frame arrival and departure time stamps, which are used to obtain high precision timing synchronization and timing distribution, as well as significantly better accuracy in performance monitoring measurements than what is obtained from pure software implementations.

The hardware time stamping functions operate in both standalone devices and in systems where multiple devices are interconnected to form a multichip system.

Figure 120 • **Supported Time Stamping Flows**



Hardware time stamping can update PTP frames coming from a standard front port without any 1588 support or from a partner device that does parts of the timing update. Each port is configured according to the partner device attached.

The modes defined in the following table comply with commonly used methods of transferring timing information in a multichip system, but are as such not specified in the IEEE 1588 standard.

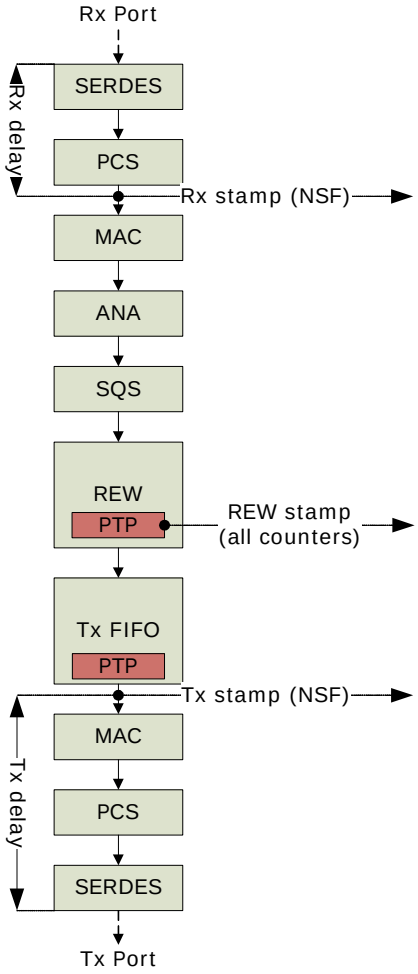
The time of day counters are generated in the DEVCPU block where there are various means of accurately updating the time. For more information, see Time of Day Generation, page 372. The output from the DEVCPU block is a set of timing parameters listed in the following table.

Table 211 • **Timing Parameters**

Parameter	Description
NSF (32 bit)	Counts number of nanoseconds since PTP was enabled in the device. Wraps at 2^{32}.
TOD_nsec (30 bit)	Counts the number of nanoseconds of current time of day. Wraps at 10^9, and will potentially be in synch with the partner devices.
TOD_secs (48 bit)	Counts number of seconds of current time of day. Wraps at 2^{48}.
NSX48	Counts number of nanoseconds. NSX48 always take the value of TOD_nsec + TOD_sec*10^9. The parameter is to be used for time stamping transfer to partner devices.
NSX32/NSX44	The lower 32 or 44 bits of NSX48.

The PTP frame flow through the device is shown in the following illustration.

Figure 121 • **Frame Flow**



The time stamps generated by the ports are samples of the NSF value. The ingress time is the time at which the PCS sees the start of frame, synchronized to the system clock domain. The correct 1588

ingress time is the time at which the first bit of the DMAC is received on the SERDES interface so a pre-configured delay is used to adjust for this in the time stamp calculations. The subtraction of the delay is done by the port modules. Delays due to barrel shifting are adjusted by a manual process, where the barrel shifter state must be read out and used when configuring the port's I/O delays. For more information, see Configuring I/O Delays, page 375.

Similarly, in the egress direction, a delay is added to the time stamps in order to move the time stamping point to when the first bit of the DMAC leaves the SerDes. This added delay takes place in the port modules.

The analyzer recognizes the PTP frames through VCAP IS2 matching and decides which operation to perform (one-step, two-step, or time-of-day-write) and which delays to add to the correction field (CF) to adjust for instance for asymmetrical path delays.

The rewriter modifies the PTP frame and can instruct the egress port module to add a delay to the correction field corresponding to the delay from the rewriting point to the transmission point. Otherwise, it can insert the original time stamp and the egress time stamp in a time stamp FIFO, which can be read by the CPU. The rewriter operation depends on the analyzer decisions and the PTP modes of the frame's ingress and egress ports.

The following table lists the PTP configuration available to the rewriter.

Table 212 • **Rewriter PTP Configuration**

Field	Description
MR	Mode of ingress port
MT	Mode of egress port
IDLY(2)	Two ingress delays optionally added to CF. Looked up for the ingress port. Format is 32 bits signed.
EDLY	One egress delay optionally added to CF. Looked up for the egress port. Only one delay is added per transfer. Format is 32 bits signed.
UDP_CSUM_DIS(4/6)	Can disable updating the UDP checksums in IPv4 and IPv6 frames.

3.31.1 One-Step Functions

The mode of a port determines how timing information is exchanged between the port and its link partner. Only the rewriter uses this information for executing the correct calculations and for deciding the right delta command to be executed in the egress port module. When the analyzer has decided to do a one-step timing update of a frame, the rewriter updates the correction field to the time at which it passes through the rewriter, and it will afterwards either prepare the PDU with timing information for the link partner, or it will ask the port module to add the time delay to the serial transmission onto the correction field. The following sections describe the PTP port modes used by the rewriter when doing one-step updates.

3.31.1.1 Front

Front ports are ports on the boundary of the local PTP system. Frames received must have their timing information updated to the time at which they left the link partner, and frames transmitted will likewise be updated to the time of transmission.

3.31.1.2 RSRV32

In a multichip system, the RSRV32 mode can be used for interconnect links. The ingress unit must update the CF field with the residence time from ingress link to any point in the ingress unit, and in the RSRV field of the PDU set the value of NSX32 at the time of CF update. As the egress unit in a multichip system has synchronized NSX32 (common clock by some means), it will be able to update the CF field with time passed from the ingress rewriter to the time of system departure.

3.31.1.3 RSRV30

Similar to the mode RSRV32, but the reference point is moved to the time of departure from the egress unit. The time value to use in the reserved bytes field only is the TOD_nsec value instead of NSX32.

3.31.1.4 ADDS48

The interconnect link in this mode carries frames where the ingress time in NSX48 format is subtracted from the CF field. Likewise, the egress unit will add the NSX48 at time of departure. This is accomplished by the following steps.

- Ingress rewriter: CF is added time from ingress port to rewriter (NSF delta)
- Ingress rewriter: CF is subtracted NSX value when doing the rewrite
- Egress rewriter: CF is added NSX value when doing the rewrite
- Egress port: CF is added time from rewriter to departure (NSF delta)

3.31.1.5 ADDS44

This mode operates as the ADDS48, only NSX44 is used instead. In this mode the egress unit must be able to see that the NSX44 has wrapped since the ingress unit did the subtraction, ie. if the NSX44 at ingress was 0xFFF.FFFF.FFFF, and at egress is 0x000.0000.0132, a larger value was subtracted than added. This is handled by the ingress unit setting CF(46) to the NSX48(44) value, and the egress unit checking the same to see if NSX48(44) has changed. In that case 2^{44} is added to CF.

3.31.1.6 MONITOR

This mode is used for an egress port where we want the frame to be updated to the time of reception only. This is used for CPU ports, where software must be able to compare the frame time with the local time stamp, in order to detect drift. It can also be used for mirroring ports where it is desired to see what time was carried on frames received from an interconnect. When a frame is transmitted on a MONITOR port, the frame will be calculated up to the ingress time stamping point. The frame will be added selected delay (IDLY/EDLY), but will not take any chip internal residence time in as a contributor to the CF.

3.31.1.7 DISABLED

A disabled PTP port is not updating PTP frames at all. This mode only applies to the destination port. If the ingress PTP mode is set to disabled, it is treated as being a FRONT port.

3.31.2 Calculation Overview

The modes of ingress and egress ports determine which PTP calculations to do. The calculations are split in an ingress half updating the frame to the time at which it passes the central rewriter and an egress half, where residence time from the rewriter is prepared and finalized in the egress port module or on the other side of the link (backplane).

In all cases the selected delay is added to CF (IDLY/EDLY).

If the ingress port mode is misconfigured as being DISABLED or MONITOR, it will be handled as a FRONT port.

The port module is through an internal preamble instructed to either do a delta update of the correction field only (CF), or to do a delta update of both the correction field and the reserved bytes field (CF_RSRV). In both cases the rewriter NSF value is sent along as a reference for the delta delay calculation in the port.

Table 213 • **Rewriter Operations for One-Step Operation**

Mode	Receiving From	Transmitting To
FRONT	CF += (NSF – IFH.STAMP)	Preamble = (CF, NSF)
RSRV30	CF += (NSEC – PDU.RSRV)	PDU.RSRV = NSEC Preamble = (CF_RSRV, NSF)
RSRV32	CF += (NSX32 – PDU.RSRV)	PDU.RSRV = NSX32

Table 213 • **Rewriter Operations for One-Step Operation *(continued)***

Mode	Receiving From	Transmitting To
ADDS44	CF += (NSX44@rew) CF += (CF_org(46)/=NSX48(44))?2^{44}:0 CF(46) = CF(47)	CF –= NSX44 CF(46) = NSX48(44)
ADDS48	CF += NSX48	CF –= NSX48
MONITOR		CF –= (NSF – IFH.TSTAMP)
DISABLED		NOP

3.31.3 Detecting Calculation Issues

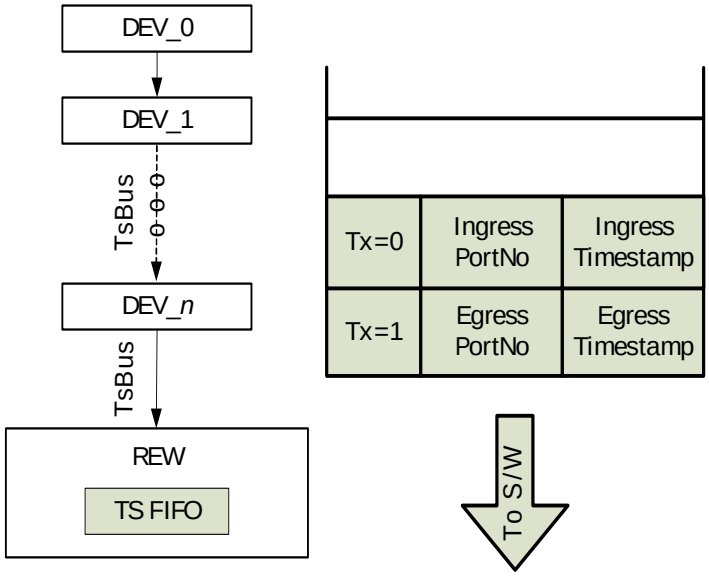
A sticky bit per port informs if frames are received with the reserved bytes field non-zero. This can be used to detect if incoming frames contain unexpected information in the reserved bytes field.

If the correction field is outside the valid range -2^{47} to $2^{47}-1$, another sticky bit is set in the port module indicating the overflow. The port module replaces in this case the final CF with the overflow indication value $2^{47}-1$. This detection is however disabled when using ADDS48 mode because the correction field in this mode rolls over regularly.

3.31.4 Two-Step Functions

The two-step PTP mode uses software as a middle stage for informing next hubs about residence times. A FIFO of time stamping events is available for software where ingress and egress time stamps as well as port numbers can be read out. The FIFO can contain 1,024 time stamp entries.

Figure 122 • **Time Stamp Bus and FIFO**



The bus passes through all port modules and terminates in a timestamp FIFO in the rewriter. If the analyzer has decided to do a two-step operation, the rewriter pushes two entries into the FIFO. The first entry contains the egress time stamp and the egress port number; the second entry contains the ingress time stamp and the ingress port number. In addition, a flag indicates whether the entry contains ingress or egress information.

When correlating frames copied to the CPU with entries in the FIFO, the ingress time stamp found in the IFH is the same as ingress time stamp found in the FIFO.

Note that the IDLY and EDLY values are not added to the CF field in the two-step case, as the frames are transferred untouched. Software must manually add these delays for the follow-up frames.

3.31.5 Time of Day Time Stamping

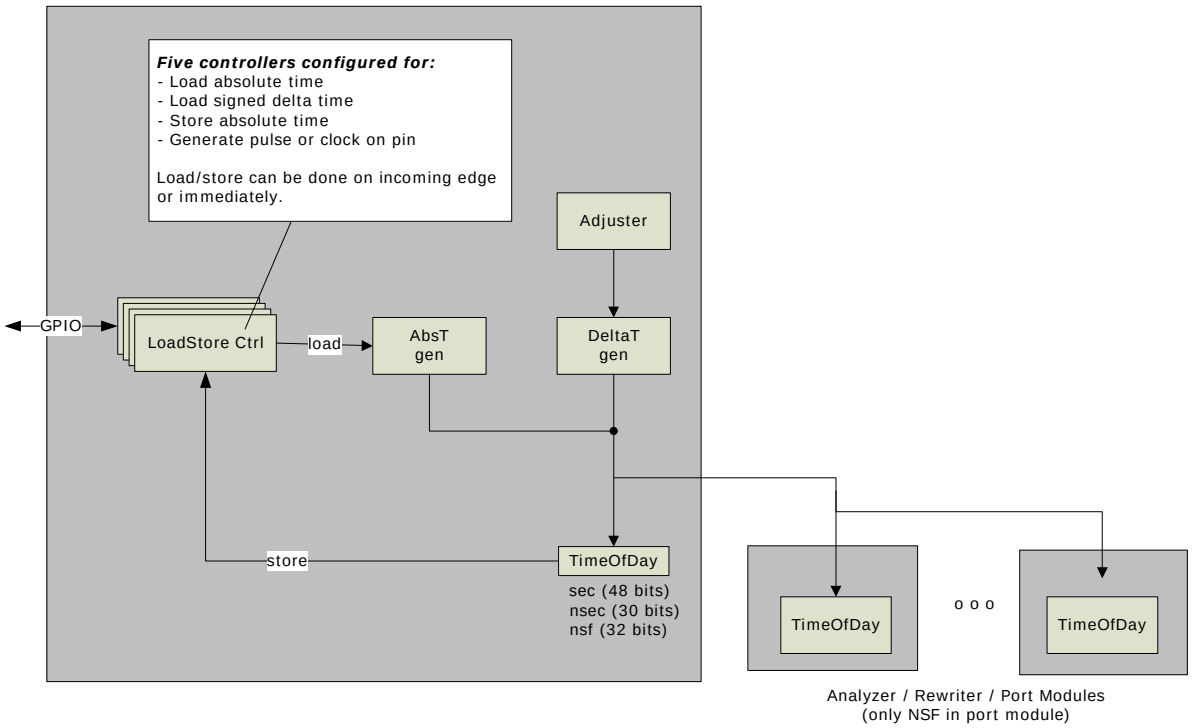
When sending out SYNC frames with full TOD, the rewriter does the TOD time stamping at the time where the frame passes through the rewriter and the port module is instructed to do a one-step update in addition. The transmitted SYNC therefore has ORG_TIME=’close to departure time’, and CF=’minor correction to that’.

```
REW: PDU.ORGTIME := (TOD_sec, TOD_nsec)
REW: Preamble := (CF, NSF)
DEV: CF += NSF@mac_Tx-ofs
```

3.31.6 Time of Day Generation

The DEVCPU has connection to four GPIOs to be used for 1588 synchronization shown in the following illustration.

Figure 123 • **Timing Distribution**



Each block using timing has a TimeOfDay instance included. These modules let time pass according to an incoming delta request, instructing it to add the nominal clock period in nanoseconds (+0/+1/–1) for each system clock cycle. This is controlled by a deltaT function in the DEVCPU, which can be configured to do regular adjustments to the time by adding a single nanosecond more or less at specified intervals. The absT function in the DEVCPU can set the full TOD time in the system. This is controlled by the LoadStore controllers.

The DEVCPU includes five LoadStore controllers. All but the last one use a designated GPIO pin on the GPIO interface. LoadStore controller 0 uses PTP_0 pin; LoadStore controller 1 uses PTP_1 pin, and so forth. Before using the LoadStore controller, the VCore-III CPU must enable the overlaid functions for the appropriate GPIO pins. For more information, see GPIO Overlaid Functions, page 439.

Each controller has CPU accessible registers with the full time of day set, and can be configured to load these into the TimeOfDay instances, or to store the current values from them. The operation can be done on a detected edge on the associated pin or immediately. The GPIO pin can also generate a clock with

configurable high and low periods set in nanoseconds using the TimeOfDay watches or it can generate a pulse at a configured time with a configurable duration and polarity.

Table 214 • **LoadStore Controller**

Pin Control Field	Function
Action	Load: Load TOD_SEC and TOD_NSEC through the absTime bus Delta: Add configured nsec to the current time of day (absT). Store: Store the current TOD_SEC,TOD_NSEC,TOD_NSF. Clock: Generate a clock or a pulse on the pin.
Sync	Execute the load/store on incoming edge, if action is LOAD or STORE. Generate a pulse instead of clock if action is CLOCK.
Inverse polarity	Falling edges are detected/generated.
TOD_sec	The 48-bit seconds of a time of day set to be loaded or stored.
TOD_nsec	The 30-bit nanoseconds of a time of day set to be loaded or stored.
TOD_NSF	The 32-bit free running timer to be stored (no load of that one)
Waveform high	Number of nanoseconds in the high period for generated clocks. Duration of pulse for generated pulse.
Waveform_low	Number of nanoseconds in the low period for generated clocks. Delay from TOD_ns = 0 for generated pulse.

In addition, the load operation can load a delta to the current time. This is done by executing a LOAD action but using the DELTA command.

Each operation generates an interrupt when executed. For the clock action, the interrupt is generated when the output switches to the active level. The interrupts from each controller can be masked and monitored by the interrupt controller in the ICPU_CFG register target.

The five controllers are completely equal in their capabilities. For concatenated distributed TC applications, two of the controllers are set up in STORE/sync mode, where incoming edges on the pin make the Time of day be stored in the controller registers.

3.31.7 Multiple PTP Time Domains

For communicating with link partners running in another TOD domain, the device includes three PTP time domains. Each port belongs to one domain only, and by default, all ports belong to time domain 0. Other time domains are accessed by configuring the time domain in the port modules, the analyzer, the rewriter, and the Loadstore controllers.

3.31.8 Register Interface to 1588 Functions

The following table lists the blocks and register configurations associated with IEEE 1588 functionality.

Table 215 • **IEEE 1588 Configuration Registers Overview**

Block	Configurations
Analyzer	PTP actions in VCAP IS2.
Rewriter	Path and asymmetry delays Enabling use of preamble Setting PTP port mode/options Accessing time stamp FIFO
Port modules	Enable rewriting. Read barrel shifter states for accuracy. I/O delays. Per port SMAC for address replacing.
CPU device	Controlling TOD generation.

3.31.8.1 VCAP IS2 Actions

In the analyzer, the VCAP IS2 actions are related to PTP operation. The REW_CMD action field determines the PTP actions in the rewriter.

Table 216 • **PTP Actions in Rewriter**

Action Bits	Description
1-0: PTP command	0: No operation. 1: One-step. 2: Two-step. 3: Time of day.
3-2: Delay command	0: Do not add any delays. 1: Add PTP_EDLY(Tx port). 2: Add PTP_IDLY1(Rx port). 3: Add PTP_IDLY2(Rx port).
5-4: Sequence number and time stamp command	0: No operation. 1: Sequence update. The rewriter contains a table of 256 sequence numbers which can be selected by the lower bits of the time stamp field from the IFH. If this command bit is set, the selected sequence number is put into the PDU, and the sequence will be incremented by one. Feature only makes sense when frame is injected by the CPU, in which case the IFH can be fully controlled by injecting with IFH. 2-3: Delay_Req/Delay_Resp processing. These modes are handled in the analyzer. This is used for automatic response generation without involving software. If bits 1-0 are set to one-step, the rewriter is requested to update the CF field to the time of reception.
6: SMAC to DMAC	If set, copy the SMAC into the DMAC of the frame.
7: Configuration to SMAC	If set, set the SMAC to the PTP SMAC configured per egress port.

3.31.8.2 Rewriter Registers

The following table lists registers associated with time stamping.

Table 217 • **Rewriter Registers**

Register	Description
PTP_MODE_CFG	Assign time domain and PTP port mode for each port.
PTP_EDLY_CFG PTP_IDLY1_CFG PTP_IDLY2_CFG	Configure ingress and egress optional delays to use per port.
PTP_SMAC_LSB PTP_SMAC_MSB	Configure the optional SMAC to replace in the PDUs.
PTP_MISC_CFG	Disable UDP checksum updating per port.
PTP_TWOSTEP_CTRL	Get next time stamp from FIFO.
PTP_TWOSTEP_STAMP	Next time stamp value.
PTP_CPUVD_MODE_CFG	Set mode and time domain for CPU and virtual device ports.
PTP_RSRV_NOT_ZERO	Sticky bit for incoming non-zero reserved bytes field.
PTP_SEQ_NO	Set or get sequence number for 256 flows.

3.31.8.3 Port Module Registers

The following table lists the port module registers associated with time stamping. The PTP_PHY_PREDICT_CFG register must be set for ports with internal PHYs.

Table 218 • **Port Module Registers**

Register	Description
PTP_CFG	Set I/O delays for port, enable PTP, and set port's time domain.
PTP_EVENTS	Sticky bit for correction field overflow.
PTP_PHY_PREDICT_CFG	Configures the timing information delivered from the internal PHYs on port 0 and port 1.
PTP_PHASE_PREDICT_CFG	Enables increased accuracy through clock phase evaluation.

3.31.9 Configuring I/O Delays

After a valid link is established and detected by the involved PCS logic, the I/O delays from the internal time stamping points to the serial line must be configured. The delays are both mode-specific and interface-specific, depending on the core clock frequency.

Ingress barrel shifter states that in 1G mode, the Rx delays must be added 0.8 ns times the value of PCS1G_LINK_STATUS.DELAY_VAR to adjust for barrel shifting state after link establishment. In 2.5G mode, the multiplier is 0.32 ns. In 100FX mode, the Rx delays must be subtracted 0.8 ns times the value of PCS_FX100_STATUS.EDGE_POS_PTP to adjust for detected data phase.

The following tables provides the Rx and Tx delay times on the ports.

Table 219 • **I/O Delays at 156.25 MHz**

Port Number	100FX Rx/Tx	1G Rx/Tx	2.5G Rx/Tx
0 – 4	69.3/79.5	43.2/44.7	
5, 6	78.2/80.8	51.7/43.9	22.0/14.1
7, 8	123.7/113.1	95.5/76.8	40.5/29.4
9, 10	123.7/113.1	95.5/76.8	39.9/29.2

Table 220 • **I/O Delays at 52 MHz**

Port Number	100FX Rx/Tx	1G Rx/Tx	2.5G Rx/Tx
0 – 4	67.3/82.9	44.0/44.2	Does not apply
5, 6	75.4/83.3	52.2/44.2	Does not apply
7, 8	116.6/116.0	98.1/75.3	Does not apply
9, 10	116.6/116.0	98.1/75.3	Does not apply

3.32 VRAP Engine

The Versatile Register Access Protocol (VRAP) engine allows external equipment to access registers in the device through any of the Ethernet ports on the device. The VRAP engine interprets incoming VRAP requests, and executes read, write, and read-modify-write commands contained in the frames. The results of the register accesses are reported back to the transmitter through automatically generated VRAP response frames.

The device supports version 1 of VRAP. All VRAP frames, both requests and responses, are standard Ethernet frames. All VRAP protocol fields are big-endian formatted.

The registers listed in the following table control the VRAP engine.

Table 221 • **VRAP Registers**

Register	Description	Replication
ANA_CL:PORT:CAPTURE_CFG. CPU_VRAP_REDIR_ENA	Enable redirection of VRAP frames to CPU.	Per port
ANA_CL::CPU_PROTO_QU_CFG. CPU_VRAP_QU	Configure the CPU extraction queue for VRAP frames.	None
QFWD:SYSTEM:FRAME_COPY_CFG. FRMC_PORT_VAL	Map VRAP CPU extraction queue to a CPU port. One CPU port must be reserved for VRAP frames.	Per CPU extraction queue
DEVCPU_QS:XTR:XTR_GRP_CFG.MODE	Enable VRAP mode for reserved CPU port.	Per CPU port
DEVCPU_QS:INJ:INJ_GRP_CFG.MODE	Enable VRAP mode injection for reserved CPU port.	Per CPU port
ASM:CFG:PORT_CFG.INJ_FORMAT_CFG	The IFH without prefix must be present for VRAP response frames.	Per port
ASM:CFG:PORT_CFG.NO_PREAMBLE_ENA	Expect no preamble in front of IFH and frame.	Per port
DEVCPU_GCB::VRAP_ACCESS_STAT	VRAP access status.	None

The VRAP engine processes incoming VRAP frames that are redirected to the VRAP CPU extraction queue by the basic classifier. For more information about the VRAP filter in the classifier, see CPU Forwarding Determination, page 112.

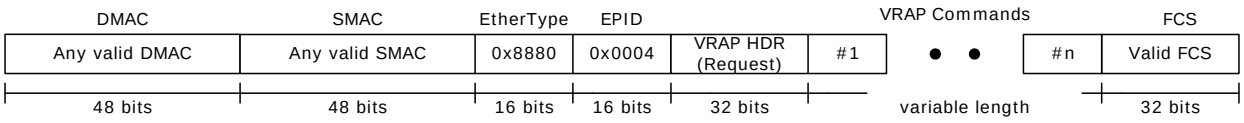
The VRAP engine is enabled by allocating one of the two CPU ports as a dedicated VRAP CPU port (DEVCPU_QS:XTR:XTR_GRP_CFG.MODE and DEVCPU_QS:INJ:INJ_GRP_CFG.MODE). The VRAP CPU extraction queue (ANA_CL::CPU_PROTO_QU_CFG.CPU_VRAP_QU) must be mapped as the only CPU extraction queue to the VRAP CPU port (QFWD:SYSTEM:FRAME_COPY_CFG.FRMC_PORT_VAL). In addition, the VRAP CPU port must enable the use of IFH without prefix and without preamble (ASM::PORT_CFG).

The complete VRAP functionality can be enabled automatically at chip startup by using special chip boot modes. For more information, see VCore-III Configurations, page 393.

3.32.1 VRAP Request Frame Format

The following illustration shows the format of a VRAP request frame.

Figure 124 • **VRAP Request Frame Format**



VRAP request frames can optionally be VLAN tagged with one VLAN tag.

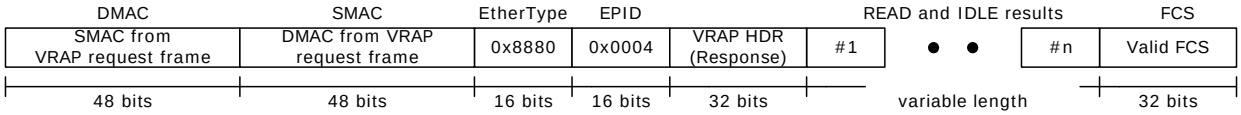
The EtherType = 0x8880 and the Ethernet Protocol Identifier (EPID) = 0x0004 identify the VRAP frames. The subsequent VRAP header is used in both request and response frames.

The VRAP commands included in the request frame are the actual register access commands. The VRAP engine supports the following five VRAP commands:

- READ: Returns the 32-bit contents of any register in the device.
- WRITE: Writes a 32-bit value to any register in the device.
- READ-MODIFY-WRITE: Does read/modify/write of any 32-bit register in the device.
- IDLE: Does not access registers; however, it is useful for padding and identification purposes.
- PAUSE: Does not access registers, but causes the VRAP engine to pause between register access.

Each of the VRAP commands are described in the following sections. Each VRAP request frame can contain multiple VRAP commands. Commands are processed sequentially starting with VRAP command 1, 2, and so on. There are no restrictions on the order or number of commands in the frame.

3.32.2 VRAP Response Frame Format



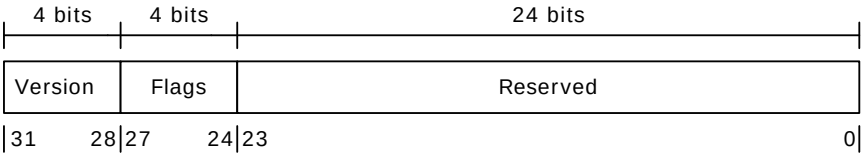
The VRAP response frame follows the VRAP request frame in terms of VLAN tagging: If the VRAP request was tagged, the VRAP response is also tagged using the same VLAN.

Only the READ and IDLE commands require data to be returned to the transmitter of the VRAP request frame. However, even if the VRAP request frame does not contain READ and IDLE commands, a VRAP response frame is generated with enough padding data (zeros) to fulfill the minimum transfer unit size.

3.32.3 VRAP Header Format

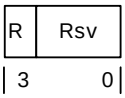
Both VRAP request and VRAP response frames contain a 32-bit VRAP header. The VRAP header is used to identify the protocol version and to differentiate between VRAP requests and VRAP response frames. The device supports VRAP version 1. A valid VRAP request frame must use Version = 1 and Flags.R = 1. Frames with an invalid VRAP header are discarded and not processed by the VRAP engine. The following illustration shows the VRAP header.

Figure 125 • **VRAP Header Format**



Version:
Set to 0x1 in VRAP response frames.
VRAP request frames are ignored if Version <> 1

Flags:
4 bits are used for Flags. Only one flag is defined:



R: 0=Request, 1=Response
Rsv: Set to 0 in VRAP response frames and ignored in VRAP request frames.

Reserved:
Set to 0 in VRAP response frames and ignored in VRAP request frames.

3.32.4 VRAP READ Command

The VRAP READ command returns the contents of any 32-bit register inside the device. The 32-bit read-data result is put into the VRAP response frame.

The READ command is 4 bytes wide and consists of one 32-bit address field, which is 32-bit aligned. The 2 least significant bits of the address set to 01. The following illustration shows the request command and the associated response result.

Figure 126 • **READ Command**

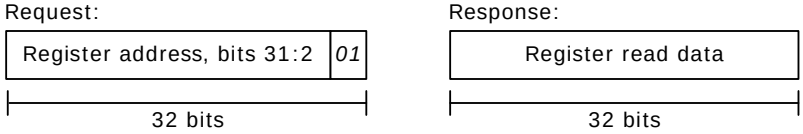
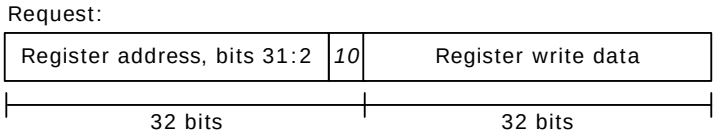


Figure 127 • **WRITE Command**

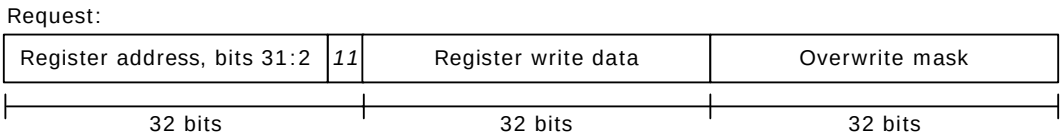


3.32.5 VRAP READ-MODIFY-WRITE Command

The READ-MODIFY-WRITE command does read/modify/write-back on any 32-bit register inside the device.

The READ-MODIFY-WRITE command is 12 bytes wide and consists of one 32-bit address field, which is 32-bit aligned. The two least significant bits of the address set to 11 followed by one 32-bit write-data field followed by one 32-bit overwrite-mask field. For bits set in the overwrite mask, the corresponding bits in the write data field are written to the register while bits cleared in the overwrite mask are untouched when writing to the register. The following illustration shows the command.

Figure 128 • **READ-MODIFY-WRITE Command**

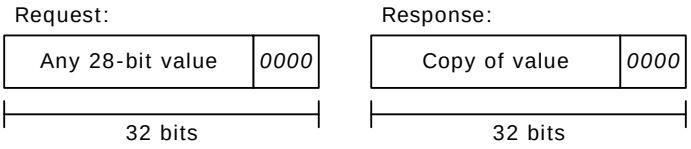


3.32.6 VRAP IDLE Command

The IDLE command does not access registers in the device. Instead it just copies itself (the entire command) into the VRAP response. This can be used for padding to fulfill the minimum transmission unit size, or an external CPU can use it to insert a unique code into each VRAP request frame so that it can separate different replies from each other.

The IDLE command is 4 bytes wide and consists of one 32-bit code word with the four least significant bits of the code word set to 0000. The following illustration depicts the request command and the associated response.

Figure 129 • **IDLE Command**

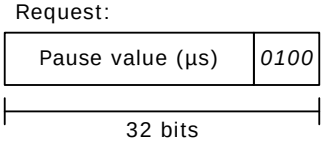


3.32.7 VRAP PAUSE Command

The PAUSE command does not access registers in the device. Instead it causes the VRAP engine to enter a wait state and remain there until the pause time has expired. This can be used to ensure sufficient delay between VRAP commands when this is needed.

The PAUSE command is 4 bytes wide and consists of one 32-bit code word with the four least significant bits of the code word set to 0100. The wait time is controlled by the 28 most significant bits of the wait command. The time unit is 1 us. The following illustration depicts the PAUSE command.

Figure 130 • **PAUSE Command**

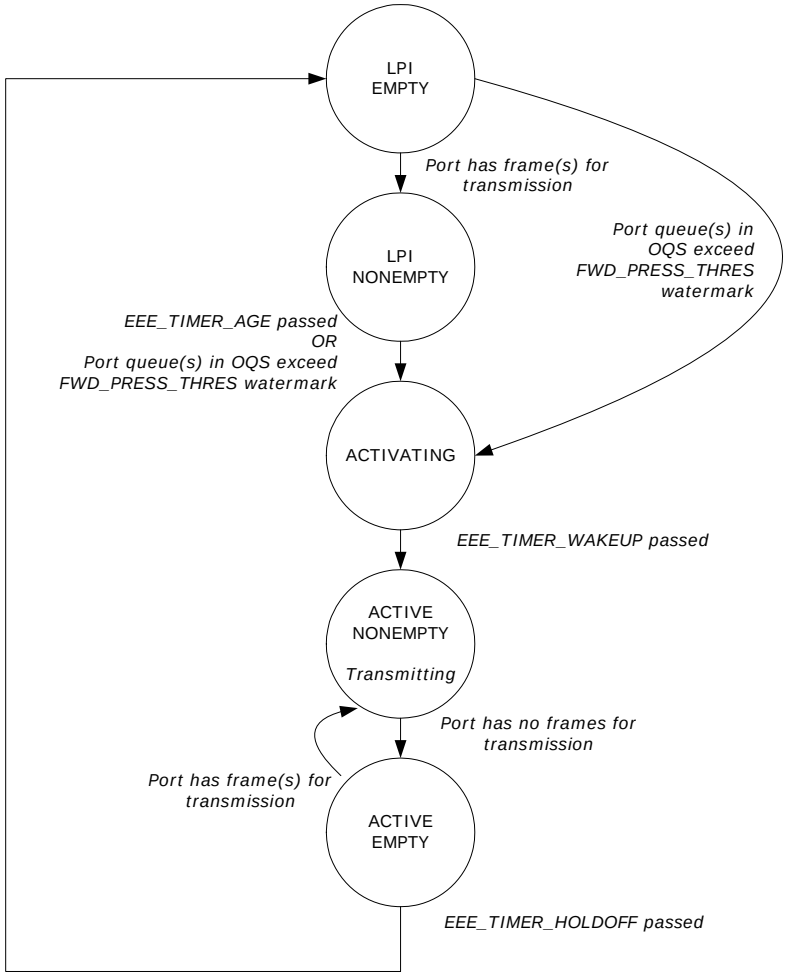


3.33 Energy Efficient Ethernet

The Ethernet ports support Energy Efficient Ethernet (EEE) to reduce power consumption when there is little or no traffic on a port.

The state diagram in the following illustration shows the EEE LPI algorithm implemented in the device.

***Figure 131* • EEE State Diagram**



- LPI_EMPTY: Low power mode. Port has no frames for transmission.
- LPI_NONEMPTY: Low power mode. Port has frames for transmission, but the number of frames is very limited (below EEE_URGENT_THRES in OQS) or EEE_TIMER_AGE has not passed yet.
- ACTIVATING: Port is being activated prior to restarting frame transmission. The timer EEE_TIMER_WAKEUP controls the length of the activation time.
- ACTIVE_NONEMPTY: Port is active and transmitting. This is the normal state for ports without EEE enabled.
- ACTIVE_EMPTY: Port is active, but currently has no frames for transmission. If frames for transmission do not become available within EEE_TIMER_HOLDOFF, then the port enters low power mode (in state LPI_EMPTY).

The state transition timing values are configured in the EEE_CFG register per port module, and the levels at which the queue system asserts forward pressure are configured in the QSYS::EEE_CFG and QSYS::EEE_THRES registers.

As illustrated, the “port has frames for transmission” condition is true if there are frames for transmission and the scheduling algorithm allows transmission of any of the available frames. So it is possible that a port enters LPI_EMPTY or stays in LPI_EMPTY, even though there are frames in one or more queues.

It is also possible, though not likely, that forward pressure is set for one or more queues when going into LPI_EMPTY. This occurs when the scheduling algorithm has decided to transmit from other queue or queues than those signalling forward pressure and the transmission of these frames causes the port leaky bucket to close.

The “forward pressure signalled for queues” condition does not look at whether the queues with forward pressure are actually allowed to transmit.

For ports that are shaped to a low bandwidth, the forward pressure watermark in OQS should be set a low value, such that it is mainly the port shaping that controls when to exit LPI.

3.34 CPU Injection and Extraction

This section provides an overview of how the CPU injects and extracts frames to and from the switch core.

The switch core forwards CPU extracted frames to eight CPU extraction queues that per queue are mapped to one of the two CPU ports (by means of QFWD:SYSTEM:FRAME_COPY_CFG[0:7]) that the CPU can access. For each CPU extraction queue, it is possible to specify the egress priority (QFWD:SYSTEM:FRAME_COPY_CFG.FRMC_QOS_ENA). For each CPU port, there is strict priority selection between the egress queues.

For injection, there are two CPU injection groups that can simultaneously forward frames into the analyzer. The CPU can access the injection and extraction groups through either a Frame DMA or direct register access.

For more information about injecting and extracting CPU traffic through front ports used as NPI port, see Internal Frame Header Placement, page 23; Setting Up a Port for Frame Injection, page 50; and Forwarding to GCPU, page 344.

3.34.1 Frame Injection

The traffic at any injection pipeline point from CPU, NPI, and AFI using the IFH (IFH.MISC.PIPELINE_ACT:= INJ_MASQ and IFH.MISC.PIPELINE_PT: = <value>).

When a frame is injected at a given pipeline point, it is logically not processed in the flow before reaching the injection point. For example, a frame injected at the ANA_CLM pipeline point bypasses VRAP detection, port Down-MEP handling, and basic classification, which requires the frame to be injected with an IFH configured with all the fields normally set by Basic classification. It also means that the frame is not accounted for in port Down-MEP loss measurement counters.

Injecting a frame at the ANA_DONE pipeline point causes it to bypass the complete analyzer and destination port must be specified in IFH (IFH.MISC.DPORT = <egress port>).

Frames can be injected as either unicast or as multicast:

- Unicast injection that allows injection into a single egress port (configured in IFH.MISC.DPORT) with all IFH.DST.ENCAP operators available in the Rewriter such as OAM and PTP detection and hardware handling (if IFH.FWD.DST_MODE is set to ENCAP).
- Multicast injection that allows injection to multiple egress ports (IFH.FWD.DST_MODE = INJECT and IFH.DST.INJECT.PORT_MASK). Because IFH.DST is used for specifying the destination set, it is not possible to use IFH.DST.ENCAP for this type of injection and all IFH.DST.ENCAP fields will default to 0. This mode bypasses the analyzer, meaning that all IFH fields used by rewriter must be given a proper value by software.

It is possible to inject frames into one of the analyzer pipeline points for ingress/Up-MEP injection or directly into one of the rewriter pipeline points for egress/Down-MEP injection.

3.34.1.1 Masqueraded Injection

The device supports masqueraded injection of frames from CPU where processing of the frame is done as if the frame was received on the masqueraded port (IFH.MISC.PIPELINE_ACT=INJ_MASQ and IFH.FWD.SRC_PORT=<masqueraded port>). A masquerade-injected frame sees the same source filter, classification, and learning as the physical port being masqueraded.

3.34.1.2 Injection Pipeline Points

The device supports specifying where in the processing flow a frame is injected by specifying a pipeline point for the frame (IFH.MISC.PIPELINE_PT0). The frame will then only be processed after the specified injection point.

For example, injecting MIP traffic into an ANA_IN_MIP pipeline point ensures that any MEP before the inner MIP position is inactive causing the MEP LM counters to not include injected MIP traffic.

3.34.2 Frame Extraction

The CPU receives frames through eight CPU extraction queues. These extraction queues can be mapped to one of the two CPU ports or any of the front ports (QFWD:SYSTEM:FRAME_COPY_CFG[0-7]). The CPU extraction queues use memory resources from the shared queue system and are subject to the thresholds and congestion rules programmed for the CPU extraction queues and the shared queue system in general.

A frame can be extracted for multiple reasons. For example, a BPDU with a new source MAC address can be extracted as both a BPDU and a learn frame with two different CPU extraction queues selected. By default, the highest CPU extraction queue number receives a copy of the frame, but it is also possible to generate multiple frame copies to all the selected CPU extraction queues (ANA_AC::CPU_CFG.ONE_CPU_COPY_ONLY_MASK).

The CPU can read frames from the CPU port in two ways:

- Reading registers in the CPU system. For more information, see Manual Extraction, page 418.
- Using the Frame DMA in the CPU system. For more information, see FDMA Extraction, page 417.

CPU extracted frames include an IFH (28-bytes) placed in front of the DMAC. The IFH contains relevant side band information about the frame, such as the frame's classification result (VLAN tag information, DSCP, QoS class) and the reason for sending the frame to the CPU. For more information about the contents of the IFH, Frame Headers, page 22.

The originating CPU extraction port number is not part of the IFH, however, a 4-byte header can be included in front of the IFH containing the CPU extraction port number (DEVCPU_QS::XTR_GRP_CFG.MODE).

3.34.3 Forwarding to CPU

Several mechanisms can be used to trigger redirection or copying of frames to the CPU. The following blocks can generate traffic to a CPU queue.

- Analyzer classifier ANA_CL
- Analyzer Layer 3 ANA_L3
- Analyzer Access Control Lists ANA_ACL
- Analyzer Layer 2 ANA_L2
- Analyzer Access Control ANA_AC
- Versatile OAM Processor VOP (both Up-MEP and Down-MEP)
- Rewriter

Frames copied or redirected to the CPU from both the rewriter and the Up-MEPs are looped and passed another time through the analyzer.

3.34.4 Automatic Frame Injection (AFI)

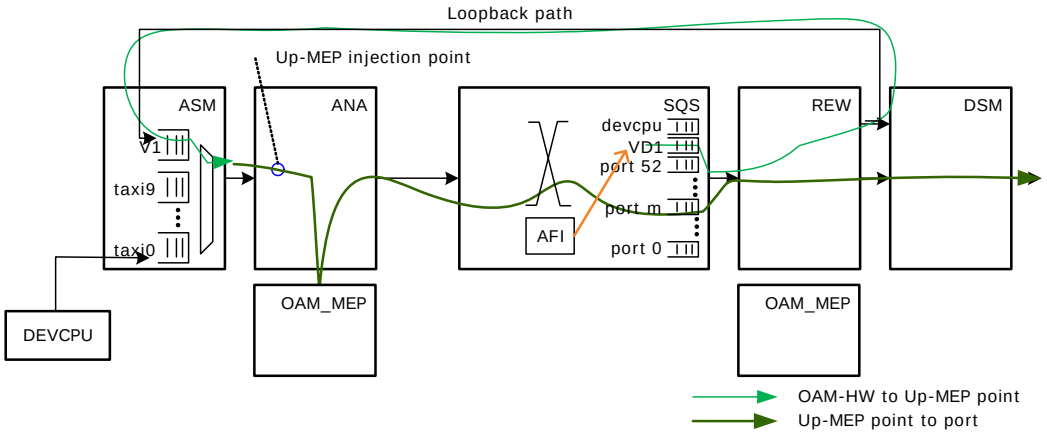
The AFI block supports repeated (automated) injection, which requires the injection frame to be sent to the AFI block with a single IFH bit set (IFH.FWD.AFI_INJ) and with an IFH header configured as for normal CPU injection.

After an injection frame is sent to the AFI block, the frame can be added for AFI injection. For more information, see Adding Injection Frame, page 320.

Frames injected from the AFI block are treated as any CPU injection, where IFH fields controls how injection occurs.

An Up-MEP injected frame from AFI must be injected on VD1, as shown in the following illustration.

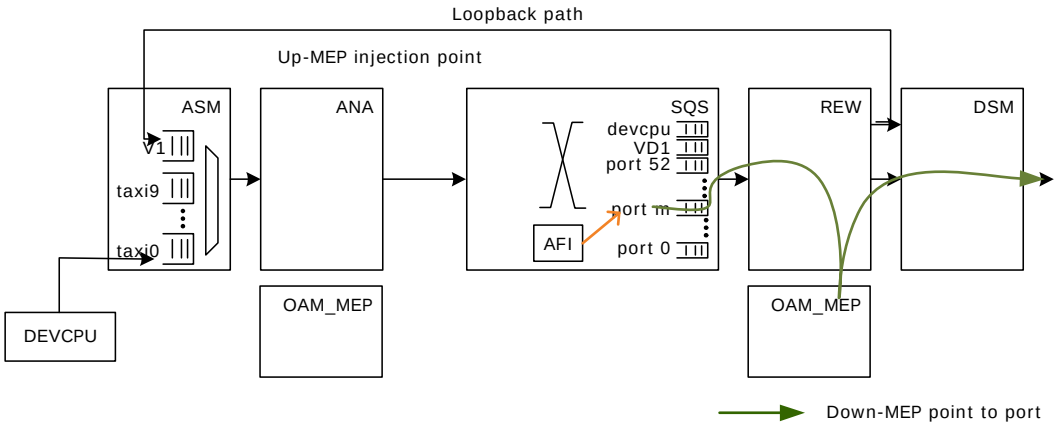
Figure 132 • **Up-MEP Injection from AFI**



An Up-MEP injected frame must be masqueraded to appear as being received at the right ingress port (by setting IFH.MISC.PIPELINE_ACT = INJ_MASQ and IFH.FWD.SRC_PORT = <masqueraded port>).

A Down-MEP injected frame must be directly injected into the right egress port as shown in the following illustration.

Figure 133 • **Down-MEP Injection from AFI**



3.35 Priority-Based Flow Control (PFC)

Priority-based flow control (PFC, IEEE 802.1Qbb) is supported on all ports for all QoS classes. The device provides independent support for transmission of pause frames and reaction to incoming pause frames, which allows asymmetric flow control configurations.

3.35.1 PFC Pause Frame Generation

The queue system monitors the congestion per ingress and egress queue. If a flow control condition is asserted, the queue system forwards the congestion status to the disassembler. The disassembler stores the congestion status per (port, priority). Based on the congestion status, the disassembler will transmit PFC frames for each port.

3.35.1.1 Queue System

The queue system has a set of dedicated PFC watermarks. If the memory consumption of a queue exceeds a PFC watermark, a flow control condition is asserted, and the updated congestion status is forwarded the disassembler. For information about the configuration of the PFC watermarks, see Priority-Based Flow Control (PFC), page 382.

3.35.1.2 Disassembler

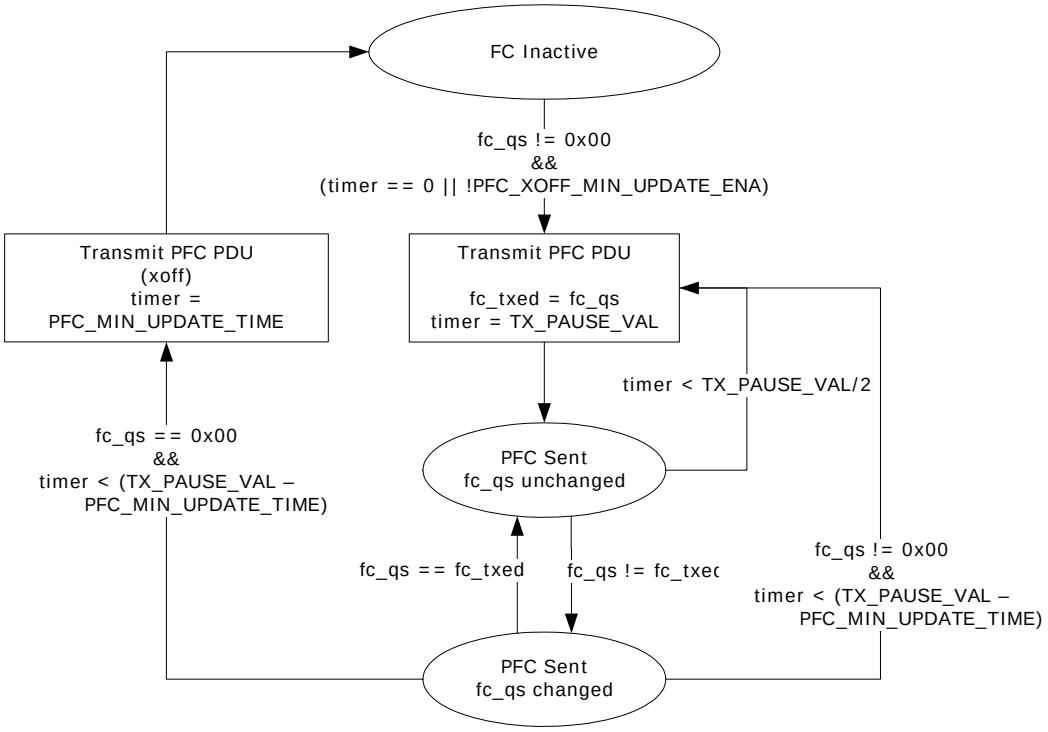
The disassembler (DSM) stores the congestion status per queue (per port, per priority) and generates PFC PDU based on the information. For more information about the available PFC configuration registers, see PFC Pause Frame Generation, page 361.

The following illustration shows the algorithm controlling the generation of PFC frames for each port. The 8-bit congestion state from the queue system is fc_qs. TX_PAUSE_VAL and PFC_MIN_UPDATE_TIME are configurable per port. The TX_PAUSE_VAL configuration is shared between LLFC and PFC.

When a priority becomes congested, a PFC PDU is generated to flow control this priority. When half of the signaled timer value has expired, and the queue is still congested, a new PFC PDU is generated. If other priorities become congested while the timer is running, then PFC_MIN_UPDATE_TIME is used to speed up the generation of the next PFC PDU. The timer is shared between all priorities per port.

Every PFC PDU signals the flow control state for all priorities of the port. In other words, the priority_enable_vector of the PFC PDU is always set to 0x00ff, independent of whether or not a priority is enabled for flow control. For disabled priorities, the pause time value is always set to 0.

Figure 134 **• PFC Generation Per Port**



fc_qs: Congestion state per priority from QS

3.35.1.3 Statistics

The number of transmitted pause frames is counted in the TX_PAUSE_CNT counter and shared with the counter for transmitted link-layer flow control pause frames. Statistics are handled locally in the DEV10G port modules. For other port module types, the assembler handles the statistics.

3.35.2 PFC Frame Reception

Received PFC frames are detected by the assembler. The status is forwarded to the queue system, which will pause the egress traffic according to the timer value specified in the pause frames.

3.35.2.1 Assembler

The assembler identifies PFC PDUs and stores the received pause time values in a set of counters. The counters are decremented according to the link speed of the port. The assembler signals a stop indication to the queue system for all (port, priority) with pause times different from 0.

For information about the PFC configuration of the assembler, see Setting Up PFC, page 52.

3.35.2.2 Queue System

The queue system stores the PFC stop indications received from the assembler per (port, priority). It pauses the transmission for all egress queues with an active PFC stop indication. The queue system does not require any PFC configuration.

3.35.2.3 Statistics

PFC pause frames are recognized as control frames and counted in the RX_UNSUP_OPCODE_CNT counter. Statistics are handled locally in the DEV10G port modules. For other port module types, the assembler handles the statistics.

3.36 Protection Switching

The following types of hardware-assisted protection switching are supported:

- Ethernet ring protection switching
- Port protection switching
- Linear protection switching for E-Lines

Both ring protection and linear protection can also be supported over a link aggregation group (LAG).

3.36.1 Ethernet Ring Protection Switching

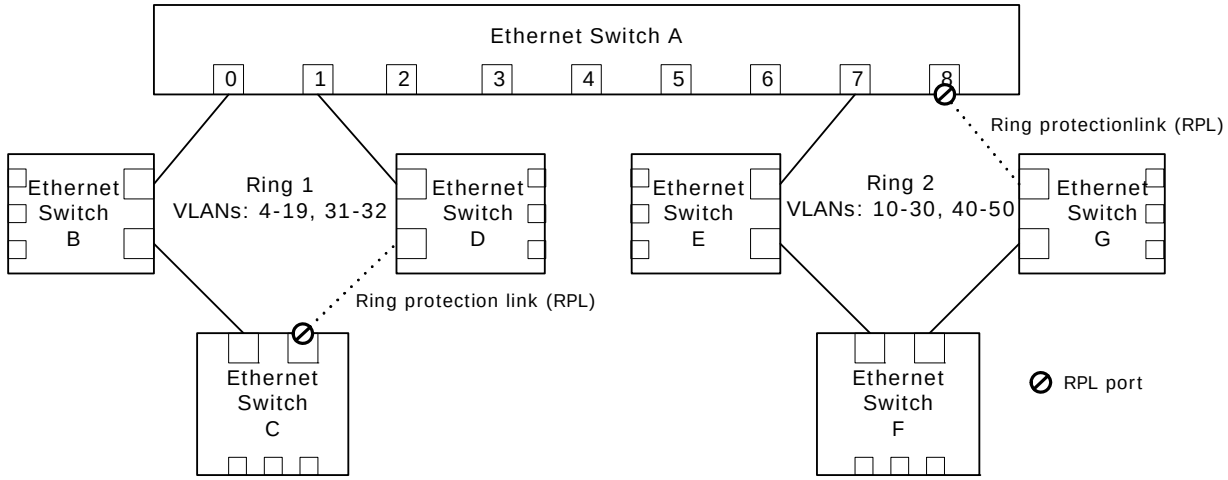
When protection switching occurs in an Ethernet ring (ITU-T G.8032), the following actions must be taken to restore connectivity.

- MAC addresses learned on the ring ports for the involved FIDs must be flushed from the MAC table. This is done using the scan functionality in the LRN block. Using ANA_L2:COMMON:SCAN_FID_CFG.SCAN_FID_VAL MAC, addresses for up to 16 FIDs can be flushed at a time.
- The RPL port must be changed to forwarding state and ports interfacing to the failed link must be changed to blocking state. For rings supporting many VLANs, this can be done efficiently using VLAN TUPE.

The following illustration shows an Ethernet ring protection example. Switch A is part of two rings, ring 1 and ring 2. Ring 1 uses 18 VIDs, VID 4-19 and 31-32, for forwarding traffic. Ring 2 uses 32 VIDs, VID 10-30 and 40-50.

To allow a VLAN TUPE command to identify the VIDs used by each of the two rings, a bit in the TUPE_CTRL field in the VLAN table is allocated for each of the two rings. In this example, ring 1 uses bit 0 in TUPE_CTRL, and ring 2 uses bit 1.

Figure 135 **• Ethernet Ring Protection Example**



The following table shows the configuration of the VLAN table of switch A. The VLAN_PORT_MASK bit shows only the value for the ring ports. For the other ports (ports 2-6), the VLAN_PORT_MASK bits are assumed to be 0.

Table 222 **• VLAN Table Configuration Before APS**

Address (VID)	TUPE_CTRL	VLAN_PORT_MASK	VLAN_FID
4-9	0x01	0x0003	4-9
10-19	0x03	0x0083	10-19
31-32	0x01	0x0003	31-32
20-30, 40-50	0x02	0x0080	20-30, 40-50

VID 4-9 and 31-32 are only used by ring 1, so only bit 0 is set in TUPE_CTRL. VLAN_PORT_MASK includes the two ring ports of ring 1, port 0, and port 1.

VID 10-19 are used by both ring 1 and ring 2, so bit 0 and bit 1 are both set in TUPE_CTRL. VLAN_PORT_MASK includes the ring ports of ring 1 as well as port 7, used by ring 2. Port 8 is not included, because it is in blocking state.

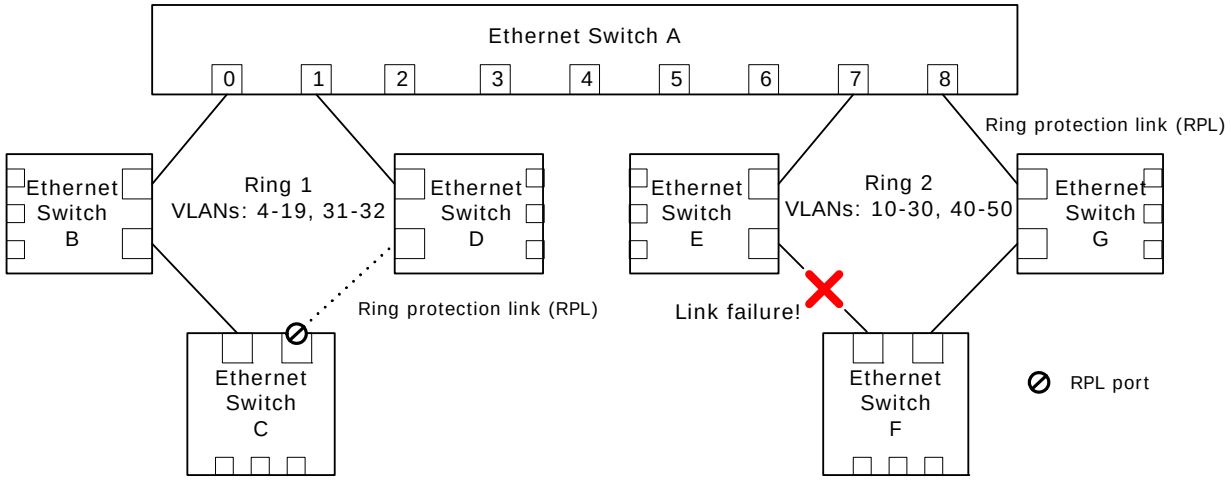
VID 20-30 and 40-50 are only used by ring 2, so only bit 1 is set in TUPE_CTRL. VLAN_PORT_MASK includes port 7.

In this example, independently learning is used for all VLANs; that is, a separate FID is used for each VID.

To also block reception of frames on any blocking ports, ANA_L3:COMMON:VLAN_FILTER_CTRL must be configured to include all ring ports. In this example ports 0,1,7 and 8 must be set in ANA_L3:COMMON:VLAN_FILTER_CTRL.

The following illustration shows the same set up as the previous illustration, but now a link in ring 2 has failed. As a result, the RPL of ring 2 must be activated, meaning port 8 of switch A must be changed to forwarding state.

Figure 136 • **Ethernet Ring with Failure**



To avoid having to perform write access to the VLAN_PORT_MASKs of each of VID 10-30 and 40-50, VLAN TUPE can be used instead. The following table lists the VLAN TUPE parameters to reconfigure the VLAN table.

Table 223 • **VLAN TUPE Command for Ring Protection Switching**

Field	Value	Comment
TUPE_CTRL_BIT_MASK	0x0002	Bits 0 and 1 of TUPE_CTRL are used as bit mask. The TUPE command is applied only to VIDs used by ring 2, so only bit 1 is set.
TUPE_CTRL_BIT_ENA	1	Enables use of TUPE_CTRL_BIT_MASK.
TUPE_START_ADDR	10	To make VLAN TUPE complete faster, only cover VID 10-50.
TUPE_END_ADDR	50	
TUPE_CMD_PORT_MASK_SET	0x0100	Enables forwarding on port 8.
TUPE_START	1	Starts VLAN TUPE. The device clears TUPE_START when TUPE is completed.

After the VLAN TUPE command is completed, the VLAN table is updated as shown in the following table.

Table 224 • **VLAN Table Configuration After APS**

Addr (VID)	TUPE_CTRL	VLAN_PORT_MASK	VLAN_FID
4-9	0x01	0x0003	4-9
10-19	0x03	0x0183	10-19
31-32	0x01	0x0003	31-32
20-30, 40-50	0x02	0x0180	20-30, 40-50

The TUPE_CTRL field is 16 bits wide, so using the described approach allows up 16 Ethernet rings with VLAN TUPE assisted protection switching. However, any unused bits in the 11-bit wide VLAN_PORT_MASK can be used in the same manner as TUPE_CTRL bits.

For Ethernet rings using only a few VIDs, there is little to gain in using VLAN TUPE and new values could instead be written directly to the VLAN_PORT_MASK of the relevant VIDs.

In addition to running the VLAN TUPE commands listed in the previous table, any MAC addresses learned on the ring ports of ring 2 must be flushed from the MAC table. In the above example, ring 2 uses 32 VIDs for forwarding traffic. Each of these use a specific FID. In this example, FID is set equal to VID.

MAC addresses for up to 16 FIDs can be flushed at a time, so to flush MAC addresses for 32 FIDs, two flush commands must be executed. The first flush command is shown in the following table. The second flush command is identical, except SCAN_FID_VAL is set to 26-30, 40-50.

Table 225 • **MAC Table Flush, First Command**

Field	Value	Comment
ANA_L2::SCAN_FID_CTRL.SCAN_FID_ENA	1	Enables use of SCAN_FID_VAL.
ANA_L2::SCAN_FID_CFG.SCAN_FID_VAL	10-25	Flushes FIDs 10-25. Flushing can be done for up to 16 discrete FID values at a time. If flushing needs to be done for less than 16 FIDs. the unused FID values must be set to 0x1FFFF.
LRN::SCAN_NEXT_CFG.FID_FILTER_ENA	1	Enables FID specific flushing.
LRN::MAC_ACCESS_CFG_0.MAC_ENTRY_FID	0x1FFFF	0x1FFFF => not used.
LRN::AUTOAGE_CFG_1.USE_PORT_FILTER_ENA	1	Enables flushing for specific ports.
ANA_L2::FILTER_LOCAL_CTRL	0x0180	Flushes port 7 and 8.
LRN::SCAN_NEXT_CFG.SCAN_NEXT_REMOVE_FOUND_ENA	1	Removes matches. In other words flushing.
LRN::COMMON_ACCESS_CTRL.MAC_TABLE_ACCESS_SHOT	1	Start flushing. The device clears MAC_TABLE_ACCESS_SHOT when flushing has completed. If desirable the device can be configured to generate an interrupt when flushing has completed. This can be used to trigger any subsequent flush commands.

Flushing MAC addresses from the MAC table usually takes longer than running VLAN TUPE. As a result, to have the protection switching complete as fast as possible, start MAC table flushing first, followed by VLAN TUPE. The two will then run in parallel, and when both have completed, the protection switching is done.

3.36.2 Linear Protection Switching for E-Line Services

For E-Line services, the device supports hardware-assisted linear protection switching. The protection entity must be pre-configured such that only a few configuration parameters need to be changed in order for the protection switching to become effective.

In the following sections, it is assumed that protection is applied to NNI and the UNI is unprotected.

A VSI or VID with only two or three ports is used to implement an E-Line service with protection switching on the NNI port. If the working and protection entity are on the same physical port, then one port is used for UNI and one port is used for NNI. If the working and protection entity are on different physical ports, then one port is used for UNI, and two ports are used for NNI.

3.36.2.1 Pre-configured Protection Configuration

The following table shows the ISDX values required for both the working and protection entity.

Table 226 • **ISDX Values for E-Line Linear Protection**

ISDX Value Name	Description
<nni isdx working>	ISDX value when coming from NNI towards UNI using the working entity.
<nni isdx protection>	ISDX value when coming from NNI towards UNI using the protection entity.
<uni isdx>	ISDX value when coming from UNI towards NNI.

The following table shows the required configuration for the IPT and PPT tables in the ANA_AC.

Table 227 • **E-Line Linear Protection Switching Configuration**

Field	Value	Comment
ANA_CL:IPT[<uni isdx>]:IPT.IPT_CFG	1	If protection entity is active, set IFH.PROT_ACTIVE = 1 to use in the rewriter. When working and protection entities are on the same physical interface this is used to select encapsulation.
ANA_CL:IPT[<nni isdx working>]:IPT.IPT_CFG	2	Discards frames received on working entity if in standby (that is, protection entity is active).
ANA_CL:IPT[<nni isdx protection>]:IPT.IPT_CFG	3	Discard frames received on protection entity if in standby (that is, working entity is active).
ANA_CL:IPT[<uni isdx>]:IPT.PPT_IDX	<ppt idx>	Index into ANA_CL:PPT. Services using the same forwarding path normally shares the same PPT_IDX value.
ANA_CL:IPT[<nni isdx working>]:IPT.PPT_IDX	<ppt idx>	Index into ANA_CL:PPT, such that frames on standby entity can be dropped.
ANA_CL:IPT[<nni isdx protection>]:IPT.PPT_IDX	<ppt idx>	Index into ANA_CL:PPT, such that frames on standby entity can be dropped.
ANA_CL:IPT[<nni isdx working>]:IPT.PROT_PIPELINE_PT ANA_CL:IPT[<nni isdx protection>]:IPT.PROT_PIPELINE_PT		Must be configured to point to where in the processing pipeline frames on the standby entity are (conceptually) dropped. This influences MEP handling and statistics counters.

For services with MAC address learning enabled, learning on the working and protection entity should be configured to learn based on an MC_IDX value instead of the source port. As a result, there is no need for flushing or moving learned MAC addresses when protection switching occurs. For each service, a unique MC_IDX must be allocated for use by the service's working and protection entities.

Table 228 • **MC_IDX MAC-Based Learning Configuration**

Field	Value	Comment
ANA_L2:ISDX[<nni isdx working>]:SERVICE_CTRL.FWD_TYPE	3	Learn MAC addresses based on MC_IDX.
ANA_L2:ISDX[<nni isdx protection>]:SERVICE_CTRL.FWD_TYPE	3	Learn MAC addresses based on MC_IDX.
ANA_L2:ISDX[<nni isdx working>]:SERVICE_CTRL.FWD_ADDR	<mc idx>	MC_IDX value for learning. Same value for both working and protection entity.
ANA_L2:ISDX[<nni isdx protection>]:SERVICE_CTRL.FWD_ADDR	<mc idx>	MC_IDX value for learning. Same value for both working and protection entity.

When MC_IDX-based MAC learning is used, it is still possible to limit the number of MAC addresses that can be learned for a given service. This is configured for the service's FID in ANA_L2:LRN_LIMIT.

The encapsulation required for forwarding using the working and protection entity is selected in the rewriter. The main mechanism for this is VCAP ES0.

The key types for VCAP ES0 lookup includes EGR_PORT and PROT_ACTIVE. Using these, in combination with ISDX/VSI/VID, it is possible to distinguish between forwarding on the working entity and forwarding on the protection entity. For MPLS-based forwarding the VCAP ES0 action provides an ENCAP_ID, which is used to point to an MPLS encapsulation. In the case of MPLS PWs, the VCAP ES0 action, as well as the rewriter mapping tables (REW:MAP_RES_A/B), can provide the inner-most LSE. For PB based forwarding the VCAP ES0 action provides the relevant VLAN tagging information.

Through configuration of VCAP ES0 it is thus possible to control the encapsulation for forwarding on working as well as protection entity.

To improve scalability in MPLS applications, such as using fewer VCAP ES0 rules, the ES0 actions ENCAP_ID_P and PROTECTION_SEL can be used for LSP protection. For the services using a given working/protection LSP pair, the ENCAP_ID in VCAP ES0 action is set to point to the encapsulation for the working LSP and the action ENCAP_ID_P can be used to select an alternative encapsulation used for the protection LSP.

When PROT_ACTIVE is set due to reconfiguration of ANA_CL:PPT:PP_CFG.STATE, the ES0 action PROTECTION_SEL controls the selected encapsulation for the protection entity.

3.36.2.2 Activating Protection Entity

When the protection entity must be activated due to fail-over, i.e. traffic must be forwarded using the protection entity instead of the working entity, the following must be done:

- ANA_CL:PPT[<ppt idx>]:PP_CFG.STATE must be configured to select the protection entity.
- If the protection entity is on a different physical port than the working entity then VLAN_PORT_MASKs in ANA_L3:VLAN must be updated for all affected services.

If VLAN_PORT_MASK needs to be changed for a large number of services, performing separate write accesses for each service may be too slow to achieve a satisfactory switch over time. To address this VLAN TUPE can be used to update the VLAN table.

Consider a scenario where 200 services are being forwarded on a working entity using port 6, and due a failure in the network, they now need to be forwarded on the protection entity on port 7 instead. Each of the services use an entry in the VLAN table. As a result, 200 VLAN table entries bit 6 of the VLAN_PORT_MASK must be cleared and bit 7 must be set instead.

Instead of writing new values to 200 VLAN_PORT_MASKs, the VLAN table entries can be pre-configured with TUPE_CTRL values to identify the entries using port 6 and port 7 as working and protection entities. A portion of the TUPE_CTRL field can be reserved for this purpose. For example, bits 8 through15 for supporting up to 256 protection groups with VLAN TUPE assisted protection switching.

If the 200 services are configured with TUPE_CTRL[15:8] set to 47, the following VLAN TUPE command can be used to efficiently update the VLAN_PORT_MASKs.

Table 229 • **VLAN TUPE Command for E-Line Linear Protection Switching**

Field	Value	Comment
TUPE_CTRL_VAL_MASK	0xff00	Bits 15:8 are used for TUPE_CTRL value field.
TUPE_CTRL_VAL	0x2f00	Bit 15:8 set to 47.
TUPE_CTRL_VAL_ENA	1	Enable use of TUPE_CTRL_VAL_MASK.
TUPE_START_ADDR	(0)	To optimize the VLAN TUPE completion time, the start address can be set to the smallest address used by the 200 services. If all VLAN table entries are processed, the start address must be set to 0.
TUPE_END_ADDR	(5119)	To optimize the VLAN TUPE completion time, the end address can be set to the largest address used by the 200 services. If all VLAN table entries are processed, the end address must be set to 5119.
TUPE_CMD_PORT_MASK_CLR	0x0040	Disable forwarding on port 6.
TUPE_CMD_PORT_MASK_SET	0x0080	Enable forwarding on port 7.
TUPE_START	1	Start VLAN TUPE. The device clears TUPE_START when VLAN TUPE has completed.

Because both E-Line linear protection switching and ring protection switching use the TUPE_CTRL field, the number of bits to be reserved for each purpose should be decided during software design. However,

unused bits in VLAN_PORT_MASK can be used to expand the TUPE_CTRL field. For more information, see Ethernet Ring Protection Switching, page 384.

3.36.3 Link Aggregation

For both ring protection and linear protection, the working/protection entity can be forwarded on a LAG, providing an additional layer of hardware assisted protection switching.

When a link within the LAG fails, the 16 port masks in ANA_AC:AGGR must be updated to avoid forwarding on the failed link.

To avoid consuming multiple entries in VCAP ES0 for services being forwarded on a LAG, REW:COMMON:PORT_CTRL.ES0_LPORT_NUM must be setup such that a common port number is used in VCAP ES0 keys, regardless of the actual LAG member chosen.

3.36.4 Port Protection Switching

The device supports several methods through which 1:1 port protection switching can be supported:

- Link Aggregation Groups. A LAG with two ports can be configured and ANA_AC:AGGR can be set to only use one of the links in the LAG.
- Disabling the standby port in ANA_L3:MSTP.
- Disabling the standby port using ANA_L3:COMMON:PORT_FWD_CTRL and ANA_L3:COMMON:PORT_LRN_CTRL. This is the recommended approach, because it interferes minimally with other features of the device.

Regardless of the method, a logical port number must be configured for the involved ports such that MAC addresses are learned on the logical port number to avoid the need for flushing MAC table entries when protection switching occurs. The logical port number is configured in ANA_CL:PORT:PORT_ID_CFG.LPORT_NUM.

REW:COMMON:PORT_CTRL.ES0_LPORT_NUM must be used to avoid consuming multiple VCAP ES0 entries. For more information, see Link Aggregation, page 390.

3.37 Clocking and Reset

There are two PLLs on the device: PLL and PLL2. The SERDES1G and SERDES6G reference clocks are driven by PLL. The switch-core clock and the VCore System clocks are driven by either PLL or PLL2.

PLL2 is disabled by default, but it can be enabled in two ways. It can be strapped to enabled by pulling strapping input REFCLK2_SEL high during device reset or it can be enabled after booting by writing HSIO::CLK_CFG.PLL2_ENA.

When PLL2 is enabled during reset, by strapping REFCLK2_SEL high, then PLL2 drives the switch-core and VCore System clocks. Otherwise, these clocks are driven by PLL. It is possible to re-configure the source of the switch core clock (including the IEEE 1588 one-second timers) by using HSIO::CLK_CFG.SWC_CLK_SRC. The VCore System clock source is not reconfigurable.

By using PLL2, the IEEE 1588 clock can be made independent of a clock recovered by Synchronous Ethernet clock.

The reference clocks for PLL and PLL2 (REFCLK_P/N and REFCLK2_P/N) are either differential or single-ended. The frequency can be 25 MHz,125 MHz, 156.25 MHz, or 250 MHz. PLL and PLL2 must be configured for the appropriate clock frequency by using strapping inputs REFCLK_CONF[2:0] and m REFCLK2_CONF[2:0].

PLL and PLL2 can be used as a recovered clock sources for Synchronous Ethernet. For information about how to configure PLL and PLL2 clock recovery, see Layer 1 Timing, page 364.

For information about protecting the VCore CPU system during a soft-reset, see Clocking and Reset, page 390.

3.37.1 Pin Strapping

Configure PLL2 reference clocks and VCore startup mode using the strapping pins on the GPIO interface. The device latches strapping pins and keeps their value when nRESET to the device is

released. After reset is released, the strapping pins are used for other functions. For more information about which GPIO pins are used for strapping, see GPIO Overlaid Functions, page 439.

By using resistors to pull the GPIOs either low or high, software can use these GPIO pins for other functions after reset has been released.

Undefined configurations are reserved and cannot be used. VCore-III configurations that enable a front port or the PCIe endpoint drive VCore_CFG[3:2] high when the front port or PCIe endpoint is ready to use.

Table 230 • **Strapping**

Pin	Description
REFCLK2_SEL	Strap high to enable PLL2. 0: PLL is source of switch core and VCore clocks. PLL2 is powered down, but can be enabled through registers. 1: PLL2 is source of switch core and VCore clocks.
REFCLK_CONF[2:0]	Reference clock frequency for PLL.
REFCLK2_CONF[2:0]	Configuration of reference clock frequency for PLL2, this is don't care if PLL2 is not enabled. 000: 125 MHz 001: 156.25 MHz 010: 250 MHz 100: 25 MHz Other values are reserved and must not be used.
VCORE_CFG[3:0]	Configuration of VCore system startup conditions. 0000: VCore-III CPU is enabled (Little Endian mode) and boots from SI (the SI slave is disabled). 0010: PCIe 1.x endpoint is enabled. Port 6 is enabled for SERDES NoAneg 1G FDX NoFC, and NPI port 4 is enabled for SERDES NoAneg 1G FDX NoFC. VRAP block is accessible through the NPI port. Au-tomatic boot of VCore-III CPU is disabled, and SI slave is enabled. 0011: PCIe 1.x endpoint is enabled. Port 0 is enabled for SERDES NoAneg 1G FDX NoFC, and NPI port 4 is enabled for SERDES NoAneg 1G FDX NoFC. VRAP block is accessible through the NPI port. Au-tomatic boot of VCore-III CPU is disabled, and SI slave is enabled. 1000: PCIe 1.x endpoint is enabled. NPI port 4 is enabled for SERDES NoAneg 1G FDX NoFC. VRAP block is accessible through the NPI port. Automatic boot of VCore-III CPU is disabled, and SI slave is enabled. 1001: PCIe 1.x endpoint is enabled. Automatic boot of VCore-III CPU is disabled, and SI slave is enabled. 1010: MIIM slave is enabled with MIIM address 0 (MIIM slave pins are overlaid on GPIOs). Automatic boot of VCore-III CPU is disabled, and SI slave is enabled. 1011: MIIM slave is enabled with MIIM address 31 (MIIM slave pins are overlaid on GPIOs). Automatic boot of VCore-III CPU is disabled, and SI slave is enabled. 1100: VCore-III CPU is enabled (Big Endian mode) and boots from SI (the SI slave is disabled). 1111: Automatic boot of VCore-III CPU is disabled, and SI slave is enabled. Other values are reserved and must not be used.

4 VCore-III System and CPU Interfaces

This section provides information about the functional aspects of blocks and interfaces related to the VCore-III on-chip microprocessor system and an external CPU system.

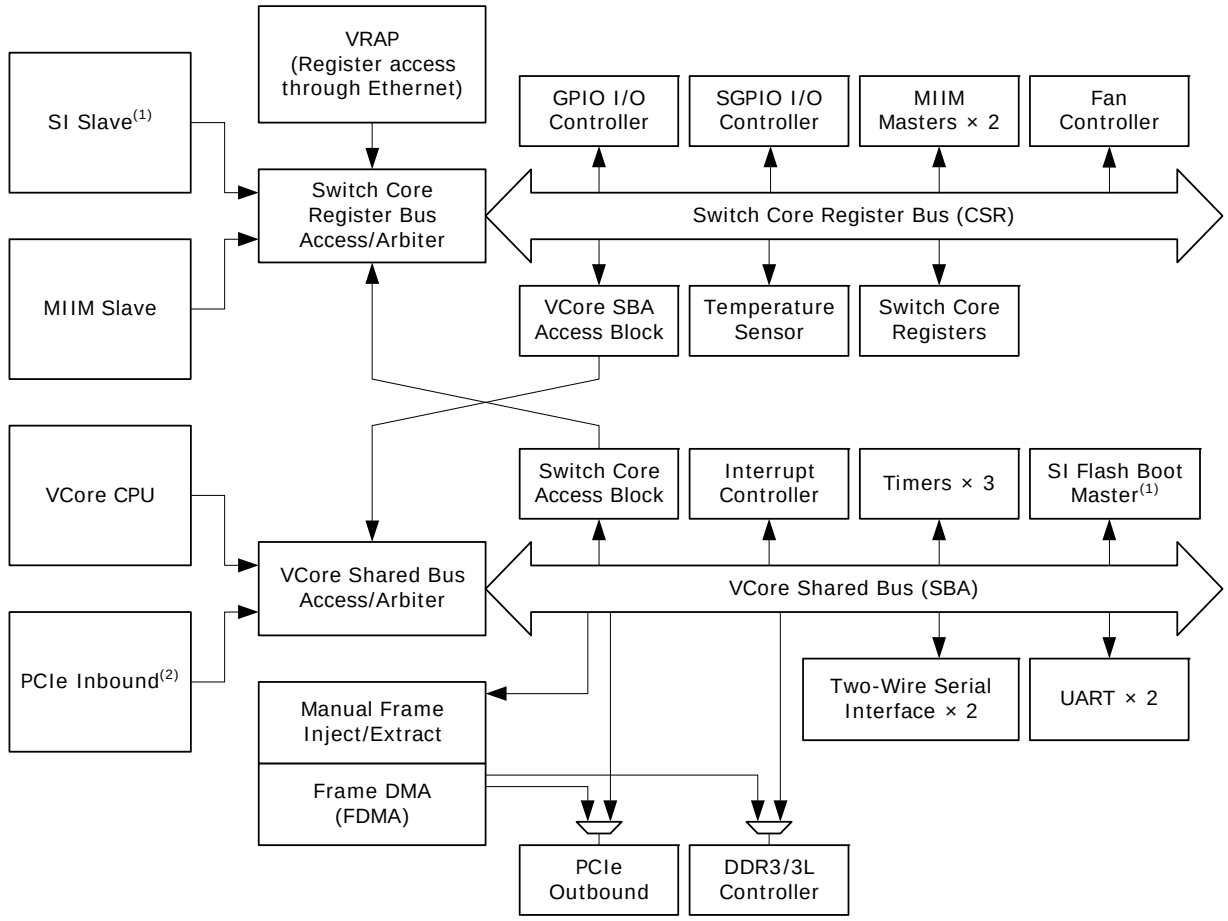
The device contains a powerful VCore-III CPU system that is based on an embedded MIPS24KEc-compatible microprocessor and a high bandwidth Ethernet Frame DMA engine. The VCore-III system can control the device independently, or it can support an external CPU, relieving the external CPU of the otherwise time consuming tasks of transferring frames, maintaining the switch core, and handling networking protocols.

When the VCore-III CPU is enabled, it either automatically boots up from serial Flash or a external CPU can manually load a code-image to the device and then start the VCore-III CPU.

An external CPU can be connected to the device through the PCIe interface, serial interface (SI), dedicated MIIM slave interface, or through Versatile Register Access Protocol (VRAP) formatted Ethernet frames using the NPI Ethernet port. Through these interfaces, the external CPU has access to the complete set of device registers and can control them without help from the VCore-III CPU if needed.

The following illustration shows the VCore-III block diagram.

Figure 137 • **VCore-III System Block Diagram**



1. When the Vcore-III CPU boots up from the SI Flash, the SI is reserved as boot interface and cannot be used by an external CPU.

2. Inbound PCIe access to BAR0 maps to VCore register space (switch core access block, interrupt controller, timers, two-wire serial master/slave, UARTs, and manual frame injection/extraction). Inbound PCIe access to BAR2 maps to the DDR3/3L memory space (DDR controller).

4.1 VCore-III Configurations

The behavior of the VCore-III system after a reset is determined by four VCore strapping pins that are overlaid on GPIO pins. The value of these GPIOs is sampled shortly after releasing reset to the device. For more information about the strapping pins, see Pin Strapping, page 390.

The strapping value determines the reset value for the ICPU_CFG::GENERAL_CTRL register. After startup, the behavior of the VCore-III system can be modified by changing some of the fields in this register. The following are common scenarios.

- After starting the device with the VCore-III CPU disabled, an external CPU can manually boot the VCore-III CPU from SI Flash by writing ICPU_CFG::GENERAL_CTRL.IF_SI_OWNER = 1, ICPU_CFG::GENERAL_CTRL.BOOT_MODE_ENA = 1, and ICPU_CFG::GENERAL_CTRL.CPU_DIS = 0. Endianess is configured by ICPU_CFG::GENERAL_CTRL.CPU_BE_ENA. Setting GENERAL_CTRL.IF_SI_OWNER disables the SI slave, so the external CPU must use another interface than SI.
- After starting the device with the VCore-III CPU disabled and loading software into DDR3/DDR3L memory, an external CPU can boot the VCore-III CPU from DDR3/DDR3L memory by writing ICPU_CFG::GENERAL_CTRL.BOOT_MODE_ENA = 0 and ICPU_CFG::GENERAL_CTRL.CPU_DIS = 0. Endianess is configured using ICPU_CFG::GENERAL_CTRL.CPU_BE_ENA.
- After automatically booting from SI Flash, the VCore-III CPU can release the SI interface and enable the SI slave by writing ICPU_CFG::GENERAL_CTRL.IF_SI_OWNER = 0. This enables SI access from an external CPU to the device. A special PCB design is required to make the serial interface work for both Flash and external CPU access.
- MIIM slave can be manually enabled by writing ICPU_CFG::GENERAL_CTRL.IF_MIIM_SLV_ENA = 1. The MIIM slave automatically takes control of the appropriate GPIO pins.

The EJTAG interface of the VCore-III CPU and the Boundary Scan JTAG controller are both multiplexed onto the JTAG interface of the device. When the JTAG_ICE_nEN pin is low, the MIPS's EJTAG controller is selected. When the JTAG_ICE_nEN pin is high, the Boundary Scan JTAG controller is selected.

4.2 Clocking and Reset

The following table lists the registers associated with reset and the watchdog timer.

Table 231 • **Clocking and Reset Configuration Registers**

Register	Description
ICPU_CFG::RESET	VCore-III reset protection scheme and initiating soft reset of the VCore-III system and/or CPU.
DEVCPU_GCB::SOFT_RST	Initiating chip-level soft reset.
ICPU_CFG::WDT	Watchdog timer configuration and status.

The VCore-III CPU runs 500 MHz, the DDR3/DDR3L controller runs 312.50 MHz, and the rest of the VCore-III system runs at 250 MHz.

The VCore-III can be soft reset by setting RESET.CORE_RST_FORCE. By default, this resets both the VCore-III CPU and the VCore-III system. The VCore-III system can be excluded from a soft reset by setting RESET.CORE_RST_CPU_ONLY; soft reset using CORE_RST_FORCE only then resets the VCore-III CPU. The frame DMA must be disabled prior to a soft reset of the VCore-III system. When CORE_RST_CPU_ONLY is set, the frame DMA, PCIe endpoint, and DDR3/DDR3L controller are not affected by a soft reset and continue to operate throughout soft reset of the VCore-III CPU.

The VCore-III system comprises all the blocks attached to the VCore Shared Bus (SBA), including the PCIe, DDR3/DDR3L, and frame DMA/injection/extraction blocks. Blocks attached to the switch core Register Bus (CSR), including VRAP, SI, and MIIM slaves, are not part of the VCore-III system reset domain. For more information about the VCore-III system blocks, see Figure 137, page 392.

The device can be soft reset by writing SOFT_RST.SOFT_CHIP_RST. The VCore-III system and CPU can be protected from a device soft reset by writing RESET.CORE_RST_PROTECT = 1 before initiating a soft reset. In this case, a chip-level soft reset is applied to all other blocks, except the VCore-III system and the CPU. When protecting the VCore-III system and CPU from a soft reset, the frame DMA must be disabled prior to a chip-level soft reset. The SERDES and PLL blocks can be protected from reset by writing to SOFT_RST.SOFT_SWC_RST instead of SOFT_CHIP_RST.

The VCore-III general purpose registers (ICPU_CFG::GPR) and GPIO alternate modes (DEVCPU_GCB::GPIO_ALT) are not affected by a soft reset. These registers are only reset when an external reset is asserted.

4.2.1 Watchdog Timer

The VCore system has a built-in watchdog timer (WDT) with a two-second timeout cycle. The watchdog timer is enabled, disabled, or reset through the WDT register. The watchdog timer is disabled by default.

After the watchdog timer is enabled, it must be regularly reset by software. Otherwise, it times out and cause a VCore soft reset equivalent to setting RESET.CORE_RST_FORCE. Improper use of the WDT.WDT_LOCK causes an immediate timeout-reset as if the watchdog timer had timed out. The WDT.WDT_STATUS field shows if the last VCore-III CPU reset was caused by WDT timeout or regular reset (possibly soft reset). The WDT.WDT_STATUS field is updated only during VCore-III CPU reset.

To enable or to reset the watchdog timer, write the locking sequence, as described in WDT.WDT_LOCK, at the same time as setting the WDT.WDT_ENABLE field.

Because watchdog timeout is equivalent to setting RESET.CORE_RST_FORCE, the RESET.CORE_RST_CPU_ONLY field also applies to watchdog initiated soft reset.

4.3 Shared Bus

The shared bus is a 32-bit address and 32-bit data bus with dedicated master and slave interfaces that interconnect all the blocks in the VCore-III system. The VCore-III CPU, PCIe inbound, and VCore SBA access block are masters on the shared bus; only they can start access on the bus.

The shared bus uses byte addresses, and transfers of 8, 16, or 32 bits can be made. To increase performance, bursting of multiple 32-bit words on the shared bus can be performed.

All slaves are mapped into the VCore-III system's 32-bit address space and can be accessed directly by masters on the shared bus. Two possible mappings of VCore-III shared bus slaves are boot mode and normal mode.

- Boot mode is active after power-up and reset of the VCore-III system. In this mode, the SI Flash boot master is mirrored into the lowest address region.
- In normal mode, the DDR3/DDR3L controller is mirrored into the lowest address region.

Changing between boot mode and normal mode is done by first writing and then reading ICPU_CFG::GENERAL_CTRL.BOOT_MODE_ENA. A change takes effect during the read.

The device supports up to 1 gigabyte (GB) of DDR3/DDR3L memory. By default, only 512 megabytes (MB) is accessible in the memory map. To accommodate more than 512 MB of memory, write ICPU_CFG::MEMCTRL_CFG.DDR_512MBYTE_PLUS = 1.

***Figure 138* • Shared Bus Memory Map**



1. To enable support 1 gigabyte (GB) of DDR3/DDR3L memory (and the associated memory map), set ICPU_CFG::MEMCTRL_CFG.DDR_512MBYTE_PLUS.

Note: When the VCore-III system is protected from a soft reset using ICPU_CFG::RESET.CORE_RST_CPU_ONLY, a soft reset does not change shared bus memory mapping. For more information about protecting the VCore-III system when using a soft reset, see Clocking and Reset, page 393.

If the boot process copies the SI Flash image to DDR3/DDR3L, and if the contents of the SI memory and the DDR3 memory are the same, software can execute from the mirrored region when swapping from boot mode to normal mode. Otherwise, software must be execute from the fixed SI Flash region when changing from boot mode to normal mode.

The Frame DMA has dedicated access to PCIe outbound and to the DDR3/DDR3L. This means that access on the SBA to other parts of the device, such as register access, does not affect Frame DMA injection/extraction performance.

4.3.1 VCore-III Shared Bus Arbitration

The following table lists the registers associated with the shared bus arbitration.

Table 232 • **Shared Bus Configuration Registers**

Register	Description
SBA::PL_CPU	Master priorities
SBA::PL_PCIE	Master priorities
SBA::PL_CSR	Master priorities
SBA::WT_EN	Enable of weighted token scheme
SBA::WT_TCL	Weighted token refresh period
SBA::WT_CPU	Token weights for weighted token scheme
SBA::WT_PCIE	Token weights for weighted token scheme
SBA::WT_CSR	Token weights for weighted token scheme

The VCore-III shared bus arbitrates between masters that want to access the bus. The default is to use a strict prioritized arbitration scheme where the VCore-III CPU has highest priority. The strict priorities can be changed using registers PL_CPU, PL_PCIE, and PL_CSR.

- *_CPU registers apply to VCore-III CPU access
- *_PCIE registers apply to inbound PCIe access
- *_CSR registers apply to VCore-III SBA access block access

It is possible to enable weighted token arbitration scheme (WT_EN). When using this scheme, specific masters can be guaranteed a certain amount of bandwidth on the shared bus. Guaranteed bandwidth that is not used is given to other masters requesting the shared bus.

When weighted token arbitration is enabled, the masters on the shared bus are grated a configurable number of tokens (WT_CPU, WT_PCIE, and WT_CSR) at the start of each refresh period. The length of each refresh period is configurable (WT_TCL). For each clock cycle that the master uses the shared bus, the token counter for that master is decremented. When all tokens are spent, the master is forced to a low priority. Masters with tokens always take priority over masters with no tokens. The strict prioritized scheme is used to arbitrate between masters with tokens and between masters without tokens.

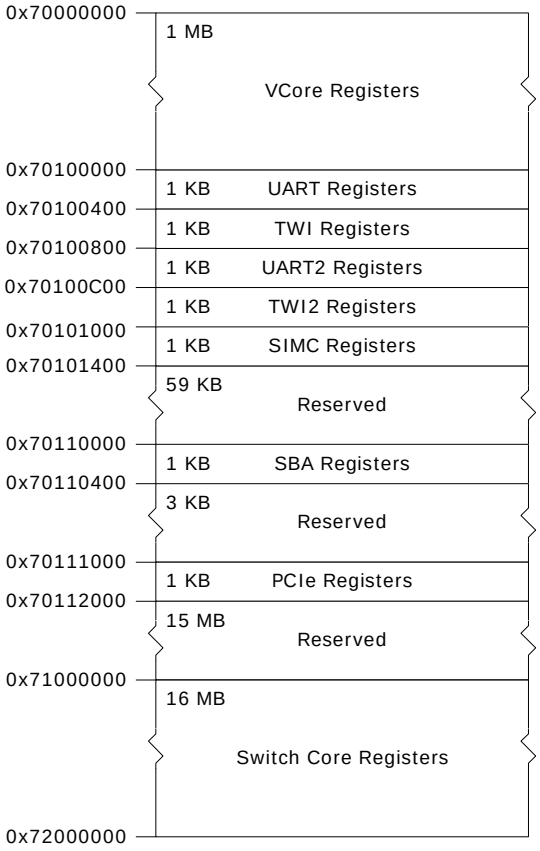
Example Guarantee That PCIe Can Get 25% Bandwidth. Configure WT_TCL to a refresh period of 2048 clock cycles; the optimal length of the refresh period depends on the scenario, experiment to find the right setting. Guarantee PCIe access in 25% of the refresh period by setting WT_PCIE to 512 (2048 × 25%). Set WT_CPU and WT_CSR to 0. This gives the VCore-III CPU and CSR unlimited tokens. Configure PCIe to highest priority by setting PL_PCIe to 15. Finally, enable the weighted token scheme by setting WT_EN to 1. For each refresh period of 2048 clock cycles, PCIe is guaranteed access to the shared bus for 512 clock cycles, because it is the highest priority master. When all the tokens are spend, it is put into the low-priority category.

4.3.2 Chip Register Region

Registers in the VCore-III domain and inside the switch core are memory mapped into the chip registers region of the shared bus memory map. All registers are 32-bit wide and must only be accessed using 32-bit reads and writes. Bursts are supported.

Writes to this region are buffered (there is a one-word write buffer). Multiple back-to-back write access pauses the shared bus until the write buffer is freed up (until the previous writes are done). Reads from this region pause the shared bus until read data is available.

***Figure 139* • Chip Registers Memory Map**



The registers in the 0x70000000 through 0x70FFFFFF region are physically located inside the VCore-III system, so read and write access to these registers is fast (done in a few clock cycles). All registers in this region are considered fast registers.

Registers in the 0x71000000 through 0x71FFFFFF region are located inside the switch core; access to registers in this range takes approximately 1 µs. The DEVCPU_ORG and DEVCPU_QS targets are special; registers inside these two targets are faster; access to these two targets takes approximately 0.1 µs.

When more than one CPU is accessing registers, the access time may be increased. For more information, see Register Access and Multimaster Systems.

Writes to the Chip Registers region is buffered (there is a one-word write buffer). Multiple back-to-back write access pauses the shared bus until the write buffer is freed up (until the previous write is done). A read access pause the shared bus until read data is available. Executing a write immediately followed by a read requires the write to be done before the read can be started.

4.3.3 SI Flash Region

Read access from the SI Flash region initiates Flash formatted read access on the SI pins of the device by means of the SI boot controller.

The SI Flash region cannot be written to. Writing to the SI interface must be implemented by using the SI master controller. For more information, see SI Master Controller, page 422. For legacy reasons, it is also possible to write to the SI interface by using the SI boot master's software "bit-banging" register interface. For more information, see SI Boot Controller, page 420.

4.3.4 DDR3/DDR3L Region

Read and write access from or to the DDR3/DDR3L region maps to access on the DDR3 interface of the device using the DDR3 controller. For more information about the DDR3/DDR3L controller and initializing DDR3/DDR3L memory, see DDR3/DDR3L Memory Controller.

4.3.5 PCIe Region

Read and write access from or to the PCIe region maps to outbound read and write access on the PCIe interface of the device by means of the PCIe endpoint controller. For more information about the PCIe endpoint controller and how to reach addresses in 32-bit and 64-bit PCIe environments, see PCIe Endpoint Controller.

4.4 VCore-III CPU

The VCore-III CPU system is based on a powerful MIPS24KEc-compatible microprocessor with 16-entry MMU, 32 kilobytes (kB) instruction, and 32 kB data caches.

This section describes how the VCore-III CPU is integrated into the VCore-III system. For more information about internal VCore-III CPU functions, such as bringing up caches, MMU, and so on, see the software board support package (BSP) at www.microsemi.com.

When the automatic boot in the little-endian or big-endian mode is enabled using the VCore-III strapping pins, the VCore-III CPU automatically starts to execute code from the SI Flash at byte address 0.

The following is a typical automatic boot sequence.

1. Speed up the boot interface. For more information, see SI boot controller.
2. Initialize the DDR3/DDR3L controller. For more information, see DDR3/DDR3L Memory Controller
3. Copy code-image from Flash to DDR3/DDR3L memory.
4. Change memory map from boot mode to normal mode, see Shared Bus.

When automatic boot is disabled, an external CPU is still able to start the VCore-III CPU through registers. For more information, see VCore Configurations.

The boot vector of the VCore-III CPU is mapped to the start of the KESEG1, which translates to physical address 0x00000000 on the VCore-III shared bus.

The VCore-III CPU interrupts are mapped to interrupt inputs 0 and 1, respectively.

4.4.1 Little Endian and Big Endian Support

The VCore-III system is constructed as a little endian system, and registers descriptions reflect little endian encoding. When big endian mode is enabled, instructions and data are byte-lane swapped just before they enter and when they leave the VCore-III CPU. This is the standard way of translating between a CPU in big endian mode and a little endian system.

In big endian mode, care must be taken when accessing parts of the memory system that is also used by users other than the VCore-III CPU. For example, device registers are written and read by the VCore-III CPU, but they are also used by the device (which sees them in little endian mode). The VCore-III BSP contains examples of code that correctly handles register access for both little and big endian mode.

Endianess of the CPU is selected by VCore-III strapping pins or by register configuration when manually booting the CPU. For more information, see VCore-III configurations.

4.4.2 Software Debug and Development

The VCore CPU has a standard MIPS EJTAG debug interface that can be used for breakpoints, loading of code, and examining memory. When the JTAG_ICE_nEN strapping pin is pulled low, the JTAG interface is attached to the EJTAG controller.

4.5 External CPU Support

An external CPU attaches to the device through the PCIe, SI, MIIM, or VRAP. Through these interfaces, an external CPU can access (and control) the device. For more information about interfaces and connections to device registers, see VCore-III system block diagram.

Inbound PCIe access is performed on the VCore Shared Bus (SBA) in the same way as ordinary VCore-III CPU access. By means of the switch core Access block it is possible to access the Switch Core Register (CSR) bus. For more information about supported PCIe BAR regions, see PCIe Endpoint Controller.

The SI, MIIM, and VRAP interfaces attach directly to the CSR. Through the VCore SBA access block, it is possible to access the VCore shared bus. For more information, see Access to the VCore Shared Bus.

The external CPU can coexist with the internal VCore-III CPU, and hardware-semaphores and interrupts are implemented for inter-CPU communication. For more information, see Mailbox and Semaphores.

4.5.1 Register Access and Multimaster Systems

There are three different groups of registers in the device:

- Switch Core
- Fast Switch Core
- VCore

The Switch Core registers and Fast Switch Core registers are separated into individual register targets and attached to the Switch Core Register bus (CSR). The Fast Switch Core registers are placed in the DEVCPU_QS and DEVCPU_ORG register targets. Access to Fast Switch Core registers is less than 0.1 µs; other Switch Core registers take no more than 1 µs to access.

The VCore registers are attached directly to the VCore shared bus. The access time to VCore registers is negligible (a few clock cycles).

Although multiple masters can access VCore registers and Switch Core registers in parallel without noticeable penalty to the access time, the following exceptions apply.

- When accessing the same Switch Core register target (for example, DEVCPU_GCB), the second master to attempt access has to wait for the first master to finish (round robin arbitration applies.) This does not apply to Fast Switch Core register targets (DEVCPU_QS and DEVCPU_ORG).
- If both the VCore-III CPU and the PCIe master are performing Switch Core Register bus (CSR) access, both need to be routed through the Switch Core Access block. The second master has to wait for the first master to finish (shared bus arbitration applies).
- If two or more SI, MIIM, or VRAP masters are performing VCore register access, they all need to go through the VCore SBA Access block. Ownership has to be resolved by use of software (for example, by using the build-in semaphores).

The most common multimaster scenario is with an active VCore-III CPU and an external CPU using either SI or VRAP. In this case, Switch Core register access to targets that are used by both CPUs may see two times the access time (no more than 2 µs).

4.5.2 Serial Interface in Slave Mode

This section provides information about the function of the serial interface in slave mode.

The following table lists the registers associated with SI slave mode.

Figure 140 **• SI Slave Mode Register**

Register	Description
DEVCPU_ORG::IF_CTRL	Configuration of endianess and bit order
DEVCPU_ORG::IF_CFGSTAT	Configuration of padding
ICPU_CFG::GENERAL_CTRL	SI interface ownership

The serial interface implements a SPI-compatible protocol that allows an external CPU to perform read and write access to register targets within the device. Endianess and bit order is configurable, and several options for high frequencies are supported.

The serial interface is available to an external CPU when the VCore-III CPU does not use the SI for Flash or external SI access. For more information, VCore-III System and CPU interfaces.

The following table lists the serial interface pins when the SI slave is configured as owner of SI interface in GENERAL_CTRL.IF_SI_OWNER.

Table 233 • **SI Slave Mode Pins**

Pin Name	I/O	Description
SI_nCS0	I	Active-low chip select
SI_CLK	I	Clock input
SI_DI	I	Data input (MOSI)
SI_DO	O	Data output (MISO)

SI_DI is sampled on rising edge of SI_CLK. SI_DO is driven on falling edge of SI_CLK. There are no requirements on the logical values of the SI_CLK and SI_DI inputs when SI_nCS is deasserted; they can be either 0 or 1. SI_DO is only driven during read access when read data is shifted out of the device.

The external CPU initiates access by asserting chip select and then transmitting one bit read/write indication, one don't care bit, 22 address bits, and 32 bits of write data (or don't care bits when reading).

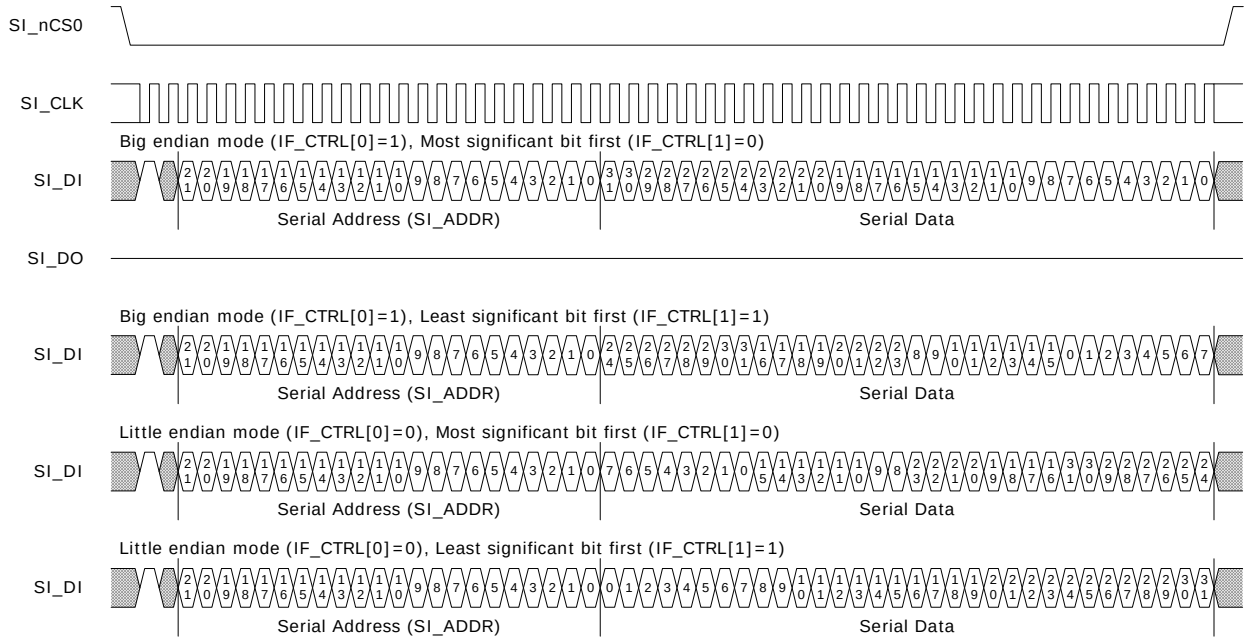
With the register address of a specific register (REG_ADDR), the SI address (SI_ADDR) is calculated as:

```
SI_ADDR = (REG_ADDR & 0x00FFFFFF)>>2
```

Data word endianess is configured through IF_CTRL[0]. The order of the data bits is configured using IF_CTRL[1].

The following illustration shows various configurations for write access. The order of the data bits during writing, as depicted, is also used when the device is transmitting data during read operations.

Figure 141 • **Write Sequence for SI**



When using the serial interface to read registers, the device needs to prepare read data after receiving the last address bit. The access time of the register that is read must be satisfied before shifting out the first bit of read data. For information about access time, see Register Access and Multimaster Systems. The external CPU must apply one of the following solutions to satisfy read access time.

- Use SI_CLK with a period of minimum twice the access time for the register target. For example, for normal switch core targets (single master):
  1/(2 × 1 µs) = 500 kHz (maximum)

- Pause the SI_CLK between shifting of serial address bit 0 and the first data bit with enough time to satisfy the access time for the register target.
- Configure the device to send out padding bytes before transmitting the read data to satisfy the access time for the register target. For example, 1 dummy byte allows enough read time for the SI clock to run up to 6 MHz in a single master system. See the following calculation.

The device is configured for inserting padding bytes by writing to IF_CFGSTAT.IF_CFG. These bytes are transmitted before the read data. The maximum frequency of the SI clock is calculated as:

```
(IF_CFGSTAT.IF_CFG × 8 – 1.5)/access-time
```

For example, for normal switch core targets (single master), 1-byte padding give(1 × 8 – 1.5) /1 µs = 6 MHz (maximum). The SI_DO output is kept tri-stated until the actual read data is transmitted.

The following illustrations show options for serial read access. The illustrations show only one mapping of read data, little endian with most significant bit first. Any of the mappings can be configured and applied to read data in the same way as for write data.

Figure 142 • **Read Sequence for SI_CLK Slow**

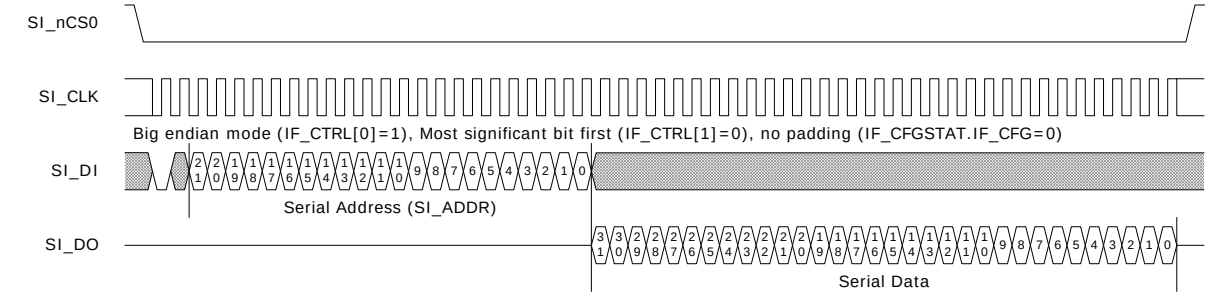


Figure 143 • **Read Sequence for SI_CLK Pause**

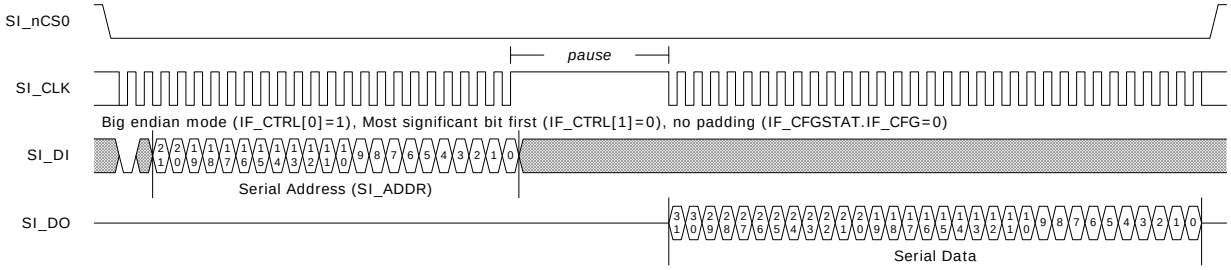
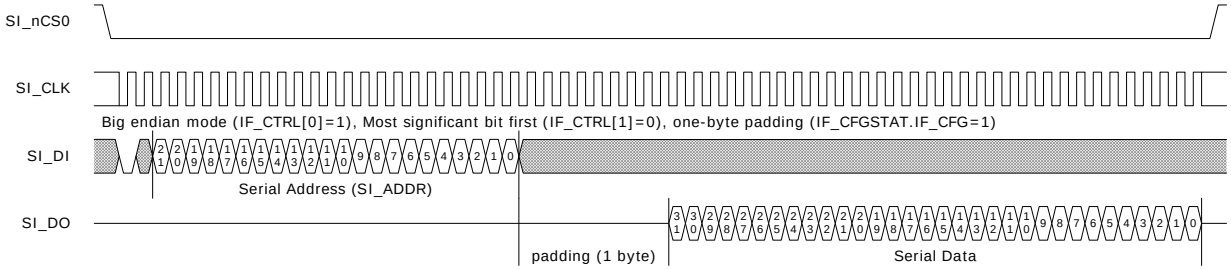


Figure 144 • **Read Sequence for One-Byte Padding**



When dummy bytes are enabled (IF_CFGSTAT.IF_CFG), the SI slave logic enables an error check that sends out 0x88888888 and sets IF_CFGSTAT.IF_STAT if the SI master does not provide enough time for register read. When using SI, the external CPU must configure the IF_CTRL register after power-up, reset, or chip-level soft reset. The IF_CTRL register is constructed so that it can be written no matter the state of the interface. For more information about constructing write data for this register, see the instructions in IF_CTRL.IF_CTRL.

4.5.3 MIIM Interface in Slave Mode

This section provides the functional aspects of the MIIM slave interface.

The MIIM slave interface allows an external CPU to perform read and write access to the device register targets. Register access is done indirectly, because the address and data fields of the MIIM protocol is less than those used by the register targets. Transfers on the MIIM interface are using the Management Frame Format protocol specified in IEEE 802.3, Clause 22.

The MIIM slave pins on the device are overlaid functions on the GPIO interface. MIIM slave mode is enabled by configuring the appropriate VCore_CFG strapping pins. For more information, see VCore-III Configurations, page 393. When MIIM slave mode is enabled, the appropriate GPIO pins are automatically overtaken. For more information about overlaid functions on the GPIOs for these signals, see GPIO Overlaid Functions, page 439.

The following table lists the pins of the MIIM slave interface.

Table 234 • **MIIM Slave Pins**

Pin Name	I/O	Description
MIIM_SLV_MDC/GPIO	I	MIIM slave clock input
MIIM_SLV_MDIO/GPIO	I/O	MIIM slave data input/output

MIIM_SLV_MDIO is sampled or changed on the rising edge of MIIM_SLV_MDC by the MIIM slave interface.

The MIIM slave can be configured to answer on one of two different PHY addresses using ICPU_CFG::GENERAL_CTRL.IF_MIIM_SLV_ADDR_SEL or the VCore_CFG strapping pins.

The MIIM slave has seven 16-bit MIIM registers defined as listed in the following table.

Table 235 • **MIIM Registers**

Register Address	Register Name	Description
0	ADDR_REG0	Bits 15:0 of the address to read or write. The address field must be formatted as word address.
1	ADDR_REG1	Bits 31:16 of the address to read or write.
2	DATA_REG0	Bits 15:0 of the data to read or write. Returns 0x8888 if a register read error occurred.
3	DATA_REG1	Bits 31:16 of the data to read or write. The read or write operation is initiated after this register is read or written. Returns 0x8888 if read while busy or a register read error occurred.
4	DATA_REG1_INCR	Bits 31:16 of data to read or write. The read or write operation is initiated after this register is read or written. When the operation is complete, the address register is incremented by one. Returns 0x8888 if read while busy or a register read error occurred.
5	DATA_REG1_INERT	Bits 31:16 of data to read or write. Reading or writing to this register does not cause a register access to be initiated. Returns 0x8888 if a register read error occurred.
6	STAT_REG	The status register gives the status of any ongoing operations. Bit 0: Busy. Set while a register read/write operation is in progress. Bit 1: Busy_rd. Busy status during the last read or write operation. Bit 2: Err. Set if a register access error occurred. Others: Reserved.

A 32-bit switch core register read or write transaction over the MIIM interface is done indirectly due to the limited data width of the MIIM frame. First, the address of the register inside the device must be set in the two 16-bit address registers of the MIIM slave using two MIIM write transactions. The two 16-bit data registers can then be read or written to access the data value of the register inside the device. Thus, it requires up to four MIIM transactions to perform a single read or write operation on a register target.

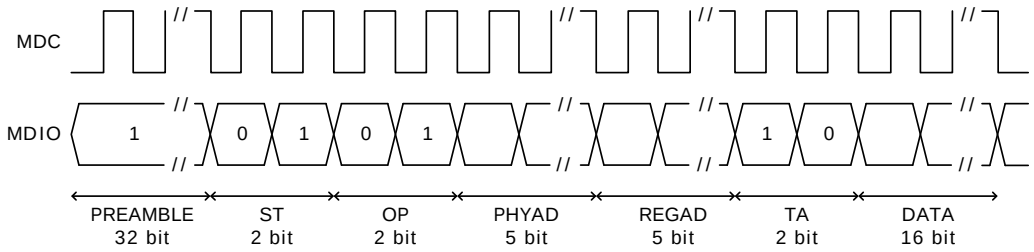
The address of the register to read/write is set in registers ADDR_REG0 and ADDR_REG1. The data to write to the register pointed to by the address in ADDR_REG0 and addr_reg1 is first written to DATA_REG0 and then to DATA_REG1. When the write transaction to DATA_REG1 is completed, the MIIM slave initiates the switch core register write.

With the register address of a specific register (REG_ADDR), the MIIM address (MIIM_ADDR) is calculated as:

MIIM_ADDR = (REG_ADDR & 0x00FFFFFF)>>2

The following illustration shows a single MIIM write transaction on the MIIM interface.

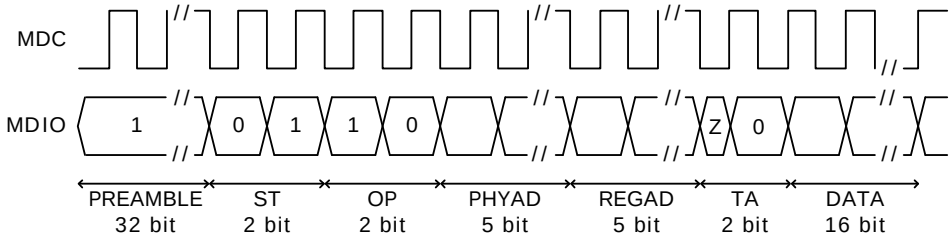
Figure 145 • **MIIM Slave Write Sequence**



A read transaction is done in a similar way. First, read the DATA_REG0 and then read the DATA_REG1. As with a write operation. The switch core register read is not initiated before the DATA_REG1 register is read. In other words, the returned read value is from the previous read transaction.

The following illustration shows a single MIIM read transaction on the MIIM interface.

Figure 146 • **MIIM Slave Read Sequence**



4.5.4 Access to the VCore Shared Bus

This section provides information about how to access the VCore shared bus (SBA) from an external CPU attached by means of the VRAP, SI, or MIIM. The following table lists the registers associated with the VCore shared bus access.

Table 236 • **VCore Shared Bus Access Registers**

Register	Description
DEVCPU_GCB::VA_CTRL	Status for ongoing access
DEVCPU_GCB::VA_ADDR	Configuration of shared bus address
DEVCPU_GCB::VA_DATA	Configuration of shared bus address
DEVCPU_GCB::VA_DATA_INCR	Data register, access increments VA_ADDR
DEVCPU_GCB::VA_DATA_INERT	Data register, access does not start new access

An external CPU perform 32-bit reads and writes to the SBA through the VCore Access (VA) registers. In the VCore-III system, a dedicated master on the shared bus handles VA access. For information about arbitration between masters on the shared bus, see VCore Shared Bus Arbitration.

The SBA address is configured in VA_ADDR. Writing to VA_DATA starts an SBA write with the 32-bit value that was written to VA_DATA. Reading from VA_DATA returns the current value of the register and

starts an SBA read access, when the read access completes, the result is automatically stored in the VA_DATA register.

The VA_DATA_INCR register behaves similar to VA_DATA, except that after starting an access, the VA_ADDR register is incremented by four (so that it points to the next word address in the SBA domain). Reading from the VA_DATA_INCR register returns the value of VA_DATA, writing to VA_DATA_INCR overwrites the value of VA_DATA.

Note: By using VA_DATA_INCR, sequential word addresses can be accessed without having to manually increment the VA_ADDR register between each access.

The VA_DATA_INERT register provides direct access to the VA_DATA value without starting access on the SBA. Reading from the VA_DATA_INERT register returns the value of VA_DATA, writing to VA_DATA_INERT overwrites the value of VA_DATA.

The VCore-III shared bus is capable of returning error-indication when illegal register regions are accessed. If a VA access results in an error-indication from the SBA, the VA_CTRL.VA_ERR field is set, and the VA_DATA is set to 0x88888888.

Note: SBA error indications only occur when non-existing memory regions or illegal registers are accessed. This will not happen during normal operation, so the VA_CTRL.VA_ERR indication is useful during debugging only.

Example Reading from ICPU_CFG::GPR[1] through the VA registers. The GPR register is the second register in the SBA VCore-III Registers region. Set VA_ADDR to 0x70000004, read once from VA_DATA (and discard the read value). Wait until VA_CTRL.VA_BUSY is cleared, then VA_DATA contains the value of the ICPU_CFG::GPR[1] register. Using VA_DATA_INERT (instead of VA_DATA) to read the data is appropriate, because this does not start a new SBA access.

4.5.4.1 Optimized Reading

VCore-III shared bus access is typically much faster than the CPU interface, which is used to access the VA registers. The VA_DATA register (VA_DATA_INCR and VA_DATA_INERT) return 0x88888888 while VA_CTRL.VA_BUSY is set. This means that it is possible to skip checking for busy between read access to SBA. For example, after initiating a read access from SBA, software can proceed directly to reading from VA_DATA, VA_DATA_INCR, or VA_DATA_INERT

- If the second read is different from 0x88888888, then the second read returned valid read data (the SBA access was done before the second read was performed).
- If the second read is equal to 0x88888888, then the VA logic may have been busy during the read and additional actions are required: First read the VA_CTRL.VA_BUSY field until the field is cleared (VA logic is not busy). Then read VA_DATA_INERT. If VA_DATA_INERT returns 0x88888888, then read-data is actually 0x88888888, continue to the next read. Otherwise, repeat the last read from VA_DATA, VA_DATA_INCR, or VA_DATA_INERT and then continue to the next read from there.

Optimized reading can be used for single read and sequential read access. For sequential reads, the VA_ADDR is only incremented on successful (non-busy) reads.

4.5.5 Mailbox and Semaphores

This section provides information about the semaphores and mailbox features for CPU to CPU communication. The following table lists the registers associated with mailbox and semaphores.

Table 237 • **Mailbox and Semaphore Registers**

Register	Description
DEVCPU_ORG::MAILBOX_SET	Atomic set of bits in the mailbox register.
DEVCPU_ORG::MAILBOX_CLR	Atomic clear of bits in the mailbox register.
DEVCPU_ORG::MAILBOX	Current mailbox state.
DEVCPU_ORG::SEMA_CFG	Configuration of semaphore interrupts.
DEVCPU_ORG::SEMA0	Taking of semaphore 0.

Table 237 • **Mailbox and Semaphore Registers *(continued)***

Register	Description
DEVCPU_ORG::SEMA1	Taking of semaphore 1.
DEVCPU_ORG::SEMA0_OWNER	Current owner of semaphore 0.
DEVCPU_ORG::SEMA1_OWNER	Current owner of semaphore 1.

The mailbox is a 32-bit register that can be set and cleared atomically using any CPU interface (including the VCore-III CPU). The MAILBOX register allows reading (and writing) of the current mailbox value. Atomic clear of individual bits in the mailbox register is done by writing a mask to MAILBOX_CLR. Atomic setting of individual bits in the mailbox register is done by writing a mask to MAILBOX_SET.

The device implements two independent semaphores. The semaphores are part of the Switch Core Register Bus (CSR) block and are accessible by means of the fast switch core registers. Semaphore ownership can be taken by interfaces attached to the CSR. That is, the VCore-III, VRAP, SI, and MIIM can be granted ownership. The VCore-III CPU and PCIe interface share the same semaphore, because they both access the CSR through the switch core access block.

Any CPU attached to an interface can attempt to take a semaphore n by reading SEMA0.SEMA0 or SEMA1.SEMA1. If the result is 1, the corresponding semaphore was successfully taken and is now owned by that interface. If the result is 0, the semaphore was not free. After a successfully taking a semaphore, all additional reads from the corresponding register will return 0. To release a semaphore write 1 to SEMA0.SEMA0 or SEMA1.SEMA1.

Note: Any interface can release semaphores; it does not have to the one that has taken the semaphore. This allows implementation of handshaking protocols.

The current status for a semaphore is available in SEMA0_OWNER.SEMA0_OWNER and SEMA1_OWNER.SEMA1_OWNER. See register description for encoding of owners.

Software interrupt is generated when semaphores are free or taken. Interrupt polarity is configured through SEMA_CFG.SEMA_INTR_POL. Semaphore 0 is hooked up to SW0 interrupt and semaphore 1 is hooked up to SW1 interrupt. For configuration of software-interrupt, see Interrupt Controller.

In addition to interrupting on semaphore state, software interrupt can be manually triggered by writing directly to the ICPU_CFG::INTR_FORCE register.

Software interrupts (SW0 and SW1) can be individually mapped to either the VCore-III CPU or to external interrupt outputs (to an external CPU).

4.6 PCIe Endpoint Controller

The device implements a single-lane PCIe 1.x endpoint that can be hooked up to any PCIe capable system. Using the PCIe interface, an external CPU can access (and control) the device. Ethernet frames can be injected and extracted using registers or the device can be configured to DMA Ethernet frames autonomously to/from PCIe memory. The device's DDR3/DDR3L memory is available through the PCIe interface, allowing the external CPU full read and write access to DDR3/DDR3L modules attached to the device. The DDR3/DDR3L controller and memory modules can be initialized either via PCIe or by the VCore-III CPU.

Both the VCore-III CPU and Frame DMA can generate PCIe read/write requests to any 32-bit or 64-bit memory region in PCIe memory space. However, it is up to software running on an external CPU to set up or communicate the appropriate PCIe memory mapping information to the device.

The defaults for the endpoints capabilities region and the extended capabilities region are listed in the registers list's description of the PCIE registers. The most important parameters are:

- Vendor (and subsystem vendor) ID: 0x101B, Microsemi Corporation
- Device ID: 0xB004, device family ID
- Revision ID: 0x00, device family revision ID
- Class Code: 0x028000, Ethernet Network controller
- Single function, non-Bridge, INTA and Message Signaled Interrupt (MSI) capable device

The device family 0xB004 covers several register-compatible devices. The software driver must determine actual device ID and revision by reading DEVCPU_GCB::CHIP_ID from the device's memory mapped registers region.

For information about base address registers, see Base Address Registers, Inbound Requests.

The IDs, class code, revision ID, and base address register setups can be customized before enabling the PCIe endpoint. However, it requires a manual bring-up procedure by software running locally on the VCore-III CPU. For more information, see Enabling the Endpoint.

The endpoint is power management capable and implements PCI Bus Power Management Interface Specification Revision 1.2. For more information, see Power Management.

The endpoint is MSI capable. Up to four 64-bit messages are supported. Messages can be generated on rising and falling edges of each of the two external VCore-III interrupt destinations. For more information, see Outbound Interrupts.

For information about all PCI Express capabilities and extended capabilities register defaults, see the PCIE region's register descriptions.

4.6.1 Accessing Endpoint Registers

The root complex accesses the PCIe endpoint's configuration registers by PCIe CfgRd/CfgWr requests. The VCore-III CPU can read configuration registers by means of the PCIE region. For more information, see Chip Register Region. The PCIE region must not be accessed when the PCIe endpoint is disabled.

The PCIe region is used during manual bring-up of PCIe endpoint. By using this region, it is possible to write most of the endpoint's read-only configuration registers, such as Vendor ID. The PCIe endpoint's read-only configuration register values must not be changed after the endpoint is enabled.

The VCore-III has a few dedicated PCIe registers in the ICPU_CFG:PCIE register group. An external CPU attached through the PCIe interface has to go through BAR0 to reach these registers.

4.6.2 Enabling the Endpoint

The PCIe endpoint is disabled by default. It can be enabled automatically or manually by either setting the VCore_CFG strapping pins or by software running in the VCore-III CPU.

The recommended approach is using VCore_CFG strapping pins, because it is fast and does not require special software running on the VCore-III CPU. The endpoint is ready approximately 50 ms after release of the device's nRESET. Until this point, the device ignores any attempts to do link training, and the PCIe output remains idle (tri-stated).

By using software running on the VCore-III CPU, it is possible to manually start the PCIe endpoint. Note that PCIe standard specifies a maximum delay from nRESET release to working PCIe endpoint so software must enable the endpoint as part of the boot process.

The root complex must follow standard PCIe procedures for bringing up the endpoint.

4.6.2.1 Manually Starting MAC/PCS and SerDes

This section provides information about how to manually start up the PCIe endpoint and customize selected configuration space parameters.

The following table lists the registers related to manually bringing up PCIe.

Table 238 • **Manual PCIe Bring-Up Registers**

Register	Description
ICPU_CFG::PCIE_CFG	Disable of automatic link initialization
HSIO::SERDES6G_COMMON_CFG	SERDES configuration
HSIO::SERDES6G_MISC_CFG	SERDES configuration
HSIO::SERDES6G_IB_CFG	SERDES configuration

Table 238 • **Manual PCIe Bring-Up Registers *(continued)***

Register	Description
HSIO::SERDES6G_IB_CFG1	SERDES configuration
HSIO::SERDES6G_OB_CFG	SERDES configuration
HSIO::SERDES6G_OB_CFG1	SERDES configuration
HSIO::SERDES6G_DES_CFG	SERDES configuration
HSIO::SERDES6G_PLL_CFG	SERDES configuration
HSIO::SERDES6G_MISC_CFG	SERDES configuration
HSIO::MCB_SERDES6G_ADDR_CFG	SERDES configuration
PCIE::DEVICE_ID_VENDOR_ID, PCIE::CLASS_CODE_REVISION_ID, PCIE::SUBSYSTEM_ID_SUBSYSTEM_VENDOR_ID	Device parameter customization
PCIE::BAR1, PCIE::BAR2	Base address register customization

Disable automatic link training for PICe endpoint.

1. PCIE_CFG.LTSSM_DIS = 1. Configure and enable PCIe SERDES for 2G5 mode.
1. SERDES6G_MISC_CFG = 0x00000031.
2. SERDES6G_OB_CFG = 0x60000171.
3. SERDES6G_OB_CFG1 = 0x000000B0.
4. SERDES6G_DES_CFG = 0x000068A6.
5. SERDES6G_IB_CFG = 0x3D57AC37.
6. SERDES6G_IB_CFG1 = 0x00110FF0.
7. SERDES6G_COMMON_CFG = 0x00024009.
8. MCB_SERDES6G_ADDR_CFG = 0x80000004 and wait until bit 31 is cleared.
9. SERDES6G_PLL_CFG = 0x00030F20.
10. MCB_SERDES6G_ADDR_CFG = 0x80000004 and wait until bit 31 is cleared.
11. Wait at least 20 ms.
12. SERDES6G_IB_CFG = 0x3D57AC3F.
13. SERDES6G_MISC_CFG = 0x00000030.
14. MCB_SERDES6G_ADDR_CFG = 0x80000004 and wait until bit 31 is cleared.
15. Wait at least 60 ms.
16. SERDES6G_IB_CFG = 0x3D57ACBF.
17. SERDES6G_IB_CFG1 = 0x00103FF0.
18. MCB_SERDES6G_ADDR_CFG = 0x80000004 and wait until bit 31 is cleared.

To optionally disable BAR1:

1. PCIE_CFG.PCIE_BAR_WR_ENA = 1
2. BAR1 = 0x00000000
3. PCIE_CFG.PCIE_BAR_WR_ENA = 0

To optionally increase BAR2 from 128 to 256 megabytes:

1. PCIE_CFG.PCIE_BAR_WR_ENA = 1
2. BAR2 = 0x0FFFFFFF
3. PCIE_CFG.PCIE_BAR_WR_ENA = 0

To optionally change selected PCIe configuration space values:

- Write Vendor ID/Device ID using DEVICE_ID_VENDOR_ID.
- Write Class Code/Revision ID using CLASS_CODE_REVISION_ID.
- Write Subsystem ID/Subsystem Vendor ID using SUBSYSTEM_ID_SUBSYSTEM_VENDOR_ID.

Enable automatic link training for PCIe endpoint

1. PCIE_CFG.LTSSM_DIS = 0

The last step enables the endpoint for link training, and the root complex will then be able to initialize the PCIe endpoint. After this the PCIE parameters must not be changed anymore.

4.6.3 Base Address Registers Inbound Requests

The device implements three memory regions. Read and write operations using the PCIe are translated directly to read and write access on the SBA. When manually bringing up the PCIe endpoint, BAR1 can be disabled. For more information, see Manually Starting MAC/PCS and SerDes.

Table 239 • **Base Address Registers**

Register	Description
BAR0, 32-bit, 32 megabytes	Chip registers region. This region maps to the Chip registers region in the SBA address space. See Chip Register Region. This region only supports 32-bit word-aligned reads and writes. Single and burst accesses are supported.
BAR1, 32-bit, 16 megabytes	SI Flash region. This region maps to the SI Flash region in the SBA address space. See SI Flash Region. This region only supports 32-bit word-aligned reads. Single and burst access is supported.
BAR2, 32-bit, 128 megabytes	DDR3/DDR3L region. This region maps to the low 128 megabytes of the DDR3/DDR3L region in the SBA address space. See DDR3/DDR3L Region.

This region support all access types.

To access the BAR1 region, a Flash must be attached to the SI interface of the device. For information about how to set up I/O timing and to program the Flash through BAR0; the Chip Registers Region, see SI Boot Controller, page 420.

To access the BAR2 region, DDR3/DDR3L modules must be present and initialized. For information about initializing the DDR3/DDR3L controller and modules through BAR0; the Chip Registers, see DDR3/DDR3L Memory Controller. During manual bring up of PCIe, the size of the BAR2 region can be changed to match the size of the attached DDR3/DDR3L modules. For more information, see Manually Starting MAC/PCS and SerDes.

4.6.4 Outbound Interrupts

The device supports both Message Signaled Interrupt (MSI) and Legacy PCI Interrupt Delivery. The root complex configures the desired mode using the MSI enable bit in the PCIe MSI Capability Register Set. For information about the VCore-III interrupt controller, see Interrupt Controller, page 451.

The following table lists the device registers associated PCIe outbound interrupts.

Table 240 • **PCIe Outbound Interrupt Registers**

Register	Description
ICPU_CFG::PCIE_INTR_COMMON_CFG	Interrupt mode and enable
ICPU_CFG::PCIE_INTR_CFG	MSI parameters

In legacy mode, one interrupt is supported; select either EXT_DST0 or EXT_DST1 using PCIE_INTR_COMMON_CFG.LEGASY_MODE_INTR_SEL. The PCIe endpoint uses Assert_INTA and Deassert_INTA messages when configured for legacy mode.

In MSI mode, both EXT_DST interrupts can be used. EXT_DST0 is configured though PCIE_INTR_CFG[0] and EXT_DST1 through PCIE_INTR_CFG[1]. Enable message generation on rising and/or falling edges in PCIE_INTR_CFG[*n*].INTR_RISING_ENA and PCIE_INTR_CFG[*n*].INTR_FALLING_ENA. Different vectors can be generated for rising and falling edges, configure these through PCIE_INTR_CFG[*n*].RISING_VECTOR_VAL and PCIE_INTR_CFG[*n*].FALLING_VECTOR_VAL. Finally, each EXT_DST interrupt must be given an appropriate traffic class via PCIE_INTR_CFG[*n*].TRAFFIC_CLASS.

After the root complex has configured the PCIe endpoint's MSI Capability Register Set and the external CPU has configured how interrupts from the VCore-III interrupt controller are propagated to PCIe, interrupts must then be enabled by setting PCIE_INTR_COMMON_CFG.PCIE_INTR_ENA.

4.6.5 Outbound Access

After the PCIe endpoint is initialized, outbound read/write access to PCIe memory space is initiated by reading or writing the SBA's PCIe DMA region.

The following table lists the device registers associated with PCIe outbound access.

Table 241 • **Outbound Access Registers**

Register	Description
ICPU_CFG::PCIESLV_SBA	Configures SBA outbound requests.
ICPU_CFG::PCIESLV_FDMA	Configures FDMA outbound requests.
PCIE::ATU_REGION	Select active region for the ATU_* registers.
PCIE::ATU_CFG1	Configures TLP fields.
PCIE::ATU_CFG2	Enable address translation.
PCIE::ATU_TGT_ADDR_LOW	Configures outbound PCIe address.
PCIE::ATU_TGT_ADDR_HIGH	Configures outbound PCIe address.

The PCIe DMA region is 1 gigabyte. Access in this region is mapped to any 1 gigabyte region in 32-bit or 64-bit PCIe memory space by using address translation. Two address translation regions are supported. The recommended approach is to configure the first region for SBA outbound access and the second region for FDMA outbound access.

Address translation works by taking bits [29:0] from the SBA/FDMA address, adding a configurable offset, and then using the resulting address to access into PCIe memory space. Offsets are configurable in steps of 64 kilobytes in the ATU_TGT_ADDR_HIGH and ATU_TGT_ADDR_LOW registers.

The software on the VCore-III CPU (or other SBA masters) can dynamically reconfigure the window as needed; however, the FDMA does not have that ability, so it must be disabled while updating the 1 gigabyte window that is set up for it.

Note: Although the SBA and FDMA both access the PCIe DMA region by addresses 0xC0000000 though 0xFFFFFFFF, the PCIe address translation unit can differentiate between these accesses and apply the appropriate translation.

4.6.5.1 Outbound SBA Translation Region Configuration

Configure PCIESLV_SBA.SBA_OFFSET = 0 and select address translation region 0 by writing ATU_REGION.ATU_IDX = 0. Set PCIE::ATU_BASE_ADDR_LOW.ATU_BASE_ADDR_LOW = 0x0000 and PCIE::ATU_LIMIT_ADDR.ATU_LIMIT_ADDR = 0x3FFF.

The following table lists the appropriate PCIe headers that must be configured before using the SBA's PCIe DMA region. Remaining header fields are automatically handled.

Table 242 • **PCIe Access Header Fields**

Header Field	Register::Fields
Attributes	ATU_CFG1.ATU_ATTR
Poisoned Data	PCIESLV_SBA.SBA_EP
TLP Digest Field Present	ATU_CFG1.ATU_TD
Traffic Class	ATU_CFG1.ATU_TC
Type	ATU_CFG1.ATU_TYPE

Table 242 • **PCIe Access Header Fields *(continued)***

Header Field	Register::Fields
Byte Enables	PCIESLV_SBA.SBA_BE
Message Code	ATU_CFG2.ATU_MSG_CODE

Configure the low address of the destination window in PCIe memory space as follows:

- Set ATU_TGT_ADDR_HIGH.ATU_TGT_ADDR_HIGH to bits [63:32] of the destination window. Set to 0 when a 32-bit address must be generated.
- Set ATU_TGT_ADDR_LOW.ATU_TGT_ADDR_LOW to bits [31:16] of the destination window. This field must not be set higher than 0xC000.

Enable address translation by writing ATU_CFG2.ATU_REGION_ENA = 1. SBA access in the PCIe DMA region is then mapped to PCIe memory space as defined by ATU_TGT_ADDR_LOW and ATU_TGT_ADDR_HIGH.

The header fields and the PCIe address fields can be reconfigured on-the-fly as needed; however, set ATU_REGION.ATU_IDX to 0 to ensure that the SBA region is selected.

4.6.5.2 Configuring Outbound FDMA Translation Region

Configure PCIESLV_FDMA.FDMA_OFFSET = 1, and select address translation region 1 by writing ATU_REGION.ATU_IDX = 1.

Set PCIE::ATU_BASE_ADDR_LOW.ATU_BASE_ADDR_LOW = 0x4000 and PCIE::ATU_LIMIT_ADDR.ATU_LIMIT_ADDR = 0x7FFF.

The FDMA PCIe header must be MRd/MWr type, reorderable, cache-coherent, and without ECRC. Remaining header fields are automatically handled.

Table 243 • **FDMA PCIe Access Header Fields**

Header Field	Register::Fields	Suggested Value
Attributes	ATU_CFG1.ATU_ATTR	Set to 2
TLP Digest Field Present	ATU_CFG1.ATU_TD	Set to 0
Traffic Class	ATU_CFG1.ATU_TC	Use an appropriate traffic class
Type	ATU_CFG1.ATU_TYPE	Set to 0

Configure low address of destination window in PCIe memory space as follows:

- Set ATU_TGT_ADDR_HIGH.ATU_TGT_ADDR_HIGH to bits [63:32] of the destination window. Set to 0 when a 32-bit address must be generated.
- Set ATU_TGT_ADDR_LOW.ATU_TGT_ADDR_LOW to bits [31:16] of the destination window. This field must not be set higher than 0xC000.

Enable address translation by writing ATU_CFG2.ATU_REGION_ENA = 1. The FDMA can be configured to make access in the 0xC0000000 - 0xFFFFFFFF address region. These accesses will then be mapped to PCIe memory space as defined by ATU_TGT_ADDR_LOW and ATU_TGT_ADDR_HIGH. The FDMA must be disabled if the address window needs to be updated.

4.6.6 Power Management

The device's PCIe endpoint supports D0, D1, and D3 device power-management states and associated link power-management states. The switch core does not automatically react to changes in the PCIe endpoint's power management states. It is, however, possible to enable a VCore-III interrupt on device

power state changes and then have the VCore-III CPU software make application-specific changes to the device operation depending on the power management state.

Table 244 • **Power Management Registers**

Register	Description
ICPU_CFG::PCIE_STAT	Current power management state
ICPU_CFG::PCIEPCS_CFG	Configuration of WAKE output and beacon
ICPU_CFG::PCIE_INTR	PCIe interrupt sticky events
ICPU_CFG::PCIE_INTR_ENA	Enable of PCIe interrupts
ICPU_CFG::PCIE_INTR_IDENT	Currently interrupting PCIe sources
ICPU_CFG::PCIE_AUX_CFG	Configuration of auxiliary power detection

Because the device does not implement a dedicated auxiliary power for the PCIe endpoint, the endpoint is operated from the VDD core power supply. Before the power management driver initializes the device, software can “force” auxiliary power detection by writing PCIE_AUX_CFG = 3, which causes the endpoint to report that it is capable of emitting Power Management Events (PME) messages in the D3c state.

The current device power management state is available using PCIE_STAT.PM_STATE. A change in this field’s value sets the PCIE_INTR.INTR_PM_STATE sticky bit. To enable this interrupt, set PCIE_INTR_ENA.INTR_PM_STATE_ENA. The current state of the PCIe endpoint interrupt towards the VCore-III interrupt controller is shown in PCIE_INTR_IDENT register (if different from zero, then interrupt is active).

The endpoint can emit PMEs if the PME_En bit is set in the PM Capability Register Set and if the endpoint is in power-down mode.

- Outbound request from either SBA or FDMA trigger PME.
- A change in status for an enabled outbound interrupt (either legacy or MSI) triggers PME. This feature can be disabled by setting ICPU_CFG::PCIE_INTR_COMMON_CFG.WAKEUP_ON_INTR_DIS.

In the D3 state, the endpoint transmits a beacon. The beacon function can be disabled and instead drive the WAKE output using the overlaid GPIO function. For more information about the overlaid function on the GPIO for this signal, see GPIO Overlaid Functions, page 439.

Table 245 • **PCIe Wake Pin**

Pin Name	I/O	Description
PCIe_WAKE/GPIO	O	PCIe WAKE output

Enable WAKE by setting PCIEPCS_CFG.BEACON_DIS. The polarity of the WAKE output is configured in PCIEPCS_CFG.WAKE_POL. The drive scheme is configured in PCIEPCS_CFG.WAKE_OE.

4.6.7 Device Reset Using PCIe

The built-in PCIe reset mechanism in the PCIe endpoint resets only the PCIe MAC. The device reset is not tied into the MAC reset. To reset the complete the device, use the following procedure.

1. Save the state of the PCIe controller registers using operating system.
2. Set DEVCPU_GCB::SOFT_RST.SOFT_CHIP_RST.
3. Wait for 100 ms.
4. Recover state of PCIe controller registers using operating system.

Setting SOFT_CHIP_RST will cause re-initialization of the device.

4.7 Frame DMA

This section describes the Frame DMA engine (FDMA). When FDMA is enabled, Ethernet frames can be extracted or injected autonomously to or from the device's DDR3/DDR3L memory and/or PCIe memory space. Linked list data structures in memory are used for injecting or extracting Ethernet frames. The FDMA generates interrupts when frame extraction or injection is done and when the linked lists needs updating.

Table 246 • **FDMA Registers**

Register	Description
DEVCPU_QS::XTR_GRP_CFG	CPU port ownership, extraction direction.
DEVCPU_QS::INJ_GRP_CFG	CPU port ownership, injection direction.
DEVCPU_QS::INJ_CTRL	Injection EOF to SOF spacing.
ASM::PORT_CFG	Enables IFH and disables FCS recalculation (for injected frames).
ICPU_CFG::FDMA_CH_CFG	Channel configuration, priorities, and so on.
ICPU_CFG::FDMA_CH_ACTIVATE	Enables channels.
ICPU_CFG::FDMA_CH_DISABLE	Disables channels.
ICPU_CFG::FDMA_CH_STAT	Status for channels.
ICPU_CFG::FDMA_CH_SAFE	Sets when safe to update channel linked lists.
ICPU_CFG::FDMA_DCB_LLP	Linked list pointer for channels.
ICPU_CFG::FDMA_DCB_LLP_PREV	Previous linked list pointer for channels.
ICPU_CFG::FDMA_CH_CNT	Software counters for channels.
ICPU_CFG::FDMA_INTR_LLP	NULL pointer event for channels.
ICPU_CFG::FDMA_INTR_LLP_ENA	Enables interrupt on NULL pointer event.
ICPU_CFG::FDMA_INTR_FRM	Frame done event for channels.
ICPU_CFG::FDMA_INTR_FRM_ENA	Enables interrupt on frame done event.
ICPU_CFG::FDMA_INTR_SIG	SIG counter incremented event for channels.
ICPU_CFG::FDMA_INTR_SIG_ENA	Enables interrupt on SIG counter event.
ICPU_CFG::MANUAL_INTR	Manual injection/extraction events.
ICPU_CFG::MANUAL_INTR_ENA	Enables interrupts on manual injection/extraction events.
ICPU_CFG::FDMA_EVT_ERR	Error event for channels.
ICPU_CFG::FDMA_EVT_ERR_CODE	Error event description.
ICPU_CFG::FDMA_INTR_ENA	Enables interrupt for channels.
ICPU_CFG::FDMA_INTR_IDENT	Currently interrupting channels.
DEVCPU_QS::XTR_FRM_PRUNING	Enables pruning of extraction frames.
ICPU_CFG::FDMA_CH_INJ_TOKEN_CNT	Injection tokens.
ICPU_CFG::FDMA_CH_INJ_TOKEN_TICK_RLD	Periodic addition of injection tokens.
ICPU_CFG::MANUAL_CFG	Configures manual injection/extraction.
ICPU_CFG::MANUAL_XTR	Memory region used for manual extraction.
ICPU_CFG::MANUAL_INJ	Memory region used for manual injection.
DEVCPU_QS::INJ_CTRL	Configures injection gap.
ICPU_CFG::FDMA_GCFG	Configures injection buffer watermark.

The FDMA implements two extraction channels per CPU port and a total of eight injection channels. Extraction channels are hard-coded per CPU port, and injection channels can be individually assigned to any CPU port.

- FDMA channel 0 corresponds to port 53 (group 0) extraction direction.
- FDMA channel 1 corresponds to port 54 (group 1) extraction direction.
- FDMA channel 2 through 9 corresponds to port 53 (group 0) injection direction when FDMA_CH_CFG[channel].CH_INJ_GRP is set to 0.
- FDMA channel 2 through 9 corresponds to port 54 (group 1) injection direction when FDMA_CH_CFG[channel].CH_INJ_GRP is set to 1.

The FDMA implements a strict priority scheme. Injection and extraction channels can be assigned individual priorities, which are used when the FDMA has to decide between servicing two or more channels. Channel priority is configured in FDMA_CH_CFG[ch].CH_PRIO. When channels have same priority, the higher channel number takes priority over the lower channel number.

When more than one injection channel is enabled for injection on the same CPU port, then priority determines which channel that is allowed to inject data. Ownership is re-arbitrated on frame boundaries.

The internal frame header is added in front of extracted frames and provides useful switching information about the extracted frames.

Injection frames requires an internal frame header for controlling injection parameters. The internal frame header is added in front of frame data. The device recalculates and overwrites the Ethernet FCS for frames that are injected via the CPU when requested by setting in the internal frame header.

For more information about the extraction and injection IFH, see Frame Headers, page 22.

The FDMA supports a manual mode where the FDMA decision logic is disabled and the FDMA takes and provides data in the order that was requested by an external master (internal or external CPU). The manual mode is a special case of normal FDMA operation where only few of the FDMA features apply. For more information about manual operation, see Manual Mode.

The following configuration must be performed before enabling FDMA extraction: Set XTR_GRP_CFG[group].MODE = 2.

The following configurations must be performed before enabling FDMA injection: Set INJ_GRP_CFG[group].MODE = 2. Set INJ_CTRL[group].GAP_SIZE = 0, set PORT_CFG[port].INJ_FORMAT_CFG = 1, set PORT_CFG[port].NO_PREAMBLE_ENA = 1, and set PORT_CFG[port].VSTAX2_AWR_ENA = 0.

4.7.1 DMA Control Block Structures

The FDMA processes linked lists of DMA Control Block Structures (DCBs). The DCBs have the same basic structure for both injection or for extraction. A DCB must be placed on a 32-bit word-aligned address in memory. Each DCB must have an associated data block that is placed on a 32-bit word aligned address in memory, the length of the data block must be a complete number of 32-bit words.

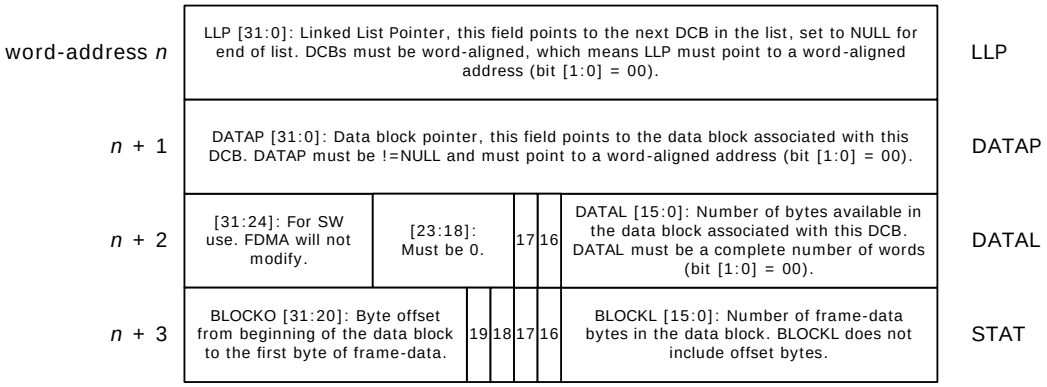
An Ethernet frame can be contained inside one data block (if the data block is big enough) or the frame can be spread across multiple data blocks. A data block never contains more than one Ethernet frame. Data blocks that contain start-of-frame have set a special bit in the DCB's status word, likewise for data blocks that contains end of frame. The FDMA stores or retrieves Ethernet frame data in network order. This means that the data at byte address (*n*) of a frame was received just before the data at byte address (*n* + 1).

Frame data inside the DCB's associated data blocks can be placed at any byte offset and have any byte length as long as byte offset and length does not exceed the size of the data block. Byte offset and length is configured using special fields inside the DCB status word. Software can specify offset both when setting up extraction and injection DCBs. Software only specifies length for injection DCBs; the FDMA automatically calculates and updates length for extraction.

Example If a DCB's status word has block offset 5 and block length 2, then the DCB's data block contains two bytes of frame data placed at second and third bytes inside the second 32-bit word of the DCB's associated data block.

DCBs are linked together by the DCB's LLP field. The last DCB in a chain must have LLP = NULL. Chains consisting of a single DCB are allowed.

Figure 147 • **FDMA DCB Layout**



STAT[16] SOF: Set to 1 if data block contains start of frame.

STAT[17] EOF: Set to 1 if data block contains end of frame.

STAT[18] ABORT: Abort indication.
Extraction: Set to 1 if the frame associated with the data block was aborted.
Injection: If set to 1 when FDMA loads the DCB, it aborts the frame associated with the data block.

STAT[19] PD: Pruned/Done indication.
Extraction: Set to 1 if the frame associated with the data block was pruned.
Injection: The FDMA set this to 1 when done processing the DCB. If set to 1 when FDMA loads the DCB, it is treated as ABORT.

DATAL[16] SIG: If set to 1 when FDMA loads the DCB, the CH_CNT_SIG counter is incremented by one.

DATAL[17] TOKEN: Token indication, only used during injection.
If set to 1, the FDMA uses one token (CH_INJ_TOKEN_CNT) when injecting the contents of the DCB. If the token counter is 0 when loading the DCB, then injection is postponed until tokens are made available.

4.7.2 Enabling and Disabling FDMA Channels

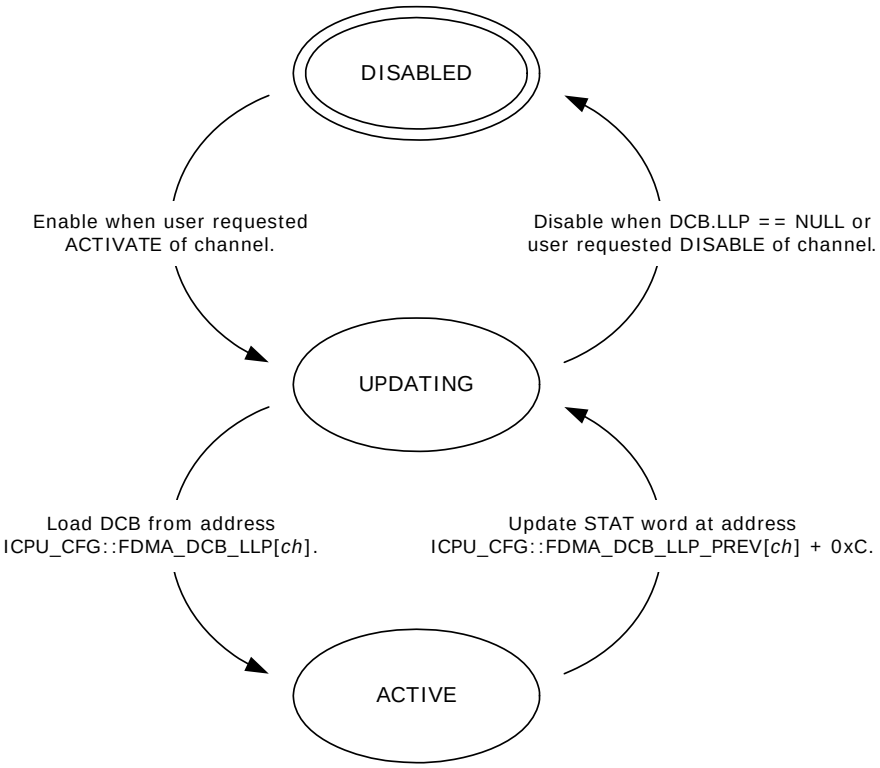
To enable a channel (ch), write a valid DCB pointer to FDMA_DCB_LLP[ch] and enable the channel by setting FDMA_CH_ACTIVATE.CH_ACTIVATE[ch]. This makes the FDMA load the DCB from memory and start either injection or extraction.

To schedule a channel for disabling, set FDMA_CH_DISABLE.CH_DISABLE[ch]. An active channel does not disable immediately. Instead it waits until the current data block is done, saves the status word, and then disables.

Channels can be in one of three states: DISABLED, UPDATING, or ACTIVE. Channels are DISABLED by default. When the channel is reading a DCB or writing the DCB status word, it is considered to be UPDATING. The status of individual channels is available in FDMA_CH_STAT.CH_STAT[ch].

The following illustration shows the FDMA channel states.

Figure 148 • **FDMA Channel States**



A channel that has FDMA_DCB_LLP[ch].DCB_LLP==NULL when going from ACTIVE to UPDATING disables itself instead of loading a new DCB. After this it can be re-enabled as previously described. Extraction channels emit an INTR_LLP-event when loading a DCB with LLP==NULL. Injection channels emit an INTR_LLP-event when saving status for a DCB that has the LLP==NULL.

Note: Extraction channel running out of DCBs during extraction is a problem that software must avoid. A hanging extraction channel will potentially be head-of-line blocking other extraction channels.

It is possible to update an active channels LLP pointer and pointers in the DCB chains. Before changing pointers software must schedule the channel for disabling (by writing FDMA_CH_DISABLE.CH_DISABLE[ch]) and then wait for the channel to set FDMA_CH_SAFE.CH_SAFE[ch]. When the pointer update is complete, soft must re-activate the channel by setting FDMA_CH_ACTIVATE.CH_ACTIVATE[ch]. Setting activate will cancel the deactivate-request, or if the channel has disabled itself in the meantime, it will re activate the channel.

Note: The address of the current DCB is available in FDMA_DCB_LLP_PREV[ch]. This information is useful when modifying pointers for an active channel. The FDMA does not reload the current DCB when re-activated, so if the LLP-field of the current DCB is modified, then software must also modify FDMA_DCB_LLP[ch].

Setting FDMA_CH_CFG[ch].DONEEOF_STOP_ENA disables an FDMA channel and emits LLP-event after saving status for DCBs that contains EOF (after extracting or injecting a complete frame). Setting FDMA_CH_CFG[ch].DONE_STOP_ENA disables an FDMA channel and emits LLP-event after saving status for any DCB.

4.7.3 Channel Counters

The FDMA implements three counters per channel: SIG, DCB, and FRM. These counters are accessible through FDMA_CH_CNT[ch].CH_CNT_SIG, FDMA_CH_CNT[ch].CH_CNT_DCB, and FDMA_CH_CNT[ch].CH_CNT_FRM, respectively. For more information about how to safely modify these counters, see the register descriptions.

- The SIG (signal) counter is incremented by one each time the FDMA loads a DCB that has the DATAL.SIG bit set to 1.

- The FRM (frame) counter is incremented by one each time the FDMA store status word for DCB that has EOF set. It is a wrapping counter that can be used for software driver debug and development. This counter does not count aborted frames.
- The DCB counter is incremented by one every time the FDMA loads a DCB from memory. It is a wrapping counter that can be used for software driver debug and development.

It is possible to enable channel interrupt whenever the SIG counter is incremented; this makes it possible for software to receive interrupt when the FDMA reaches certain points in a DCB chain.

4.7.4 FDMA Events and Interrupts

Each FDMA channel can generate four events: LLP-event, FRM-event, SIG-event, and ERR-event. These events cause a bit to be set in FDMA_INTR_LLP.INTR_LLP[ch], FDMA_INTR_FRM.INTR_FRM[ch], FDMA_INTR_SIG.INTR_SIG[ch], and FDMA_EVT_ERR.EVT_ERR[ch], respectively.

- LLP-event occurs when an extraction channel loads a DCB that has LLP = NULL or when an injection channel writes status for a DCB that has LLP=NULL. LLP-events are also emitted from channels that have FDMA_CH_CFG[ch].DONEEOF_STOP_ENA or FDMA_CH_CFG[ch].DONE_STOP_ENA set. For more information, see Enabling and Disabling FDMA Channels.
- FRM-event is indicated when an active channel loads a new DCB and the previous DCB had EOF. The FRM-event is also indicated for channels that are disabled after writing DCB status with EOF.
- SIG-event is indicated whenever the FDMA_CH_CNT[ch].CH_CNT_SIG counter is incremented. The SIG (signal) counter is incremented when loading a DCB that has the DATAL.SIG bit set.
- ERR-event is an error indication that is set for a channel if it encounters an unexpected problem during normal operation. This indication is implemented to ease software driver debugging and development. A channel that encounters an error will be disabled, depending on the type of error the channel state may be too corrupt for it to be restarted without system reset. When an ERR-event occurs, the FDMA_EVT_ERR_CODE.EVT_ERR_CODE shows the exact reason for an ERR-event. For more information about the errors that are detected, see FDMA_EVT_ERR_CODE.EVT_ERR_CODE.

Each of the events (LLP, FRM, SIG) can be enabled for channel interrupt through FDMA_INTR_LLP_ENA.INTR_LLP_ENA[ch], FDMA_INTR_FRM_ENA.INTR_FRM_ENA[ch], and FDMA_INTR_SIG.INTR_SIG[ch] respectively. The ERR event is always enabled for channel interrupt.

The highest numbered extraction channel supports two additional non-sticky events related to manual extraction: XTR_SOF_RDY-event and XTR_ANY_RDY-event. An active event causes the following fields to be set: MANUAL_INTR.INTR_XTR_SOF_RDY and MANUAL_INTR.INTR_XTR_ANY_RDY respectively. For more information, see Manual Extraction.

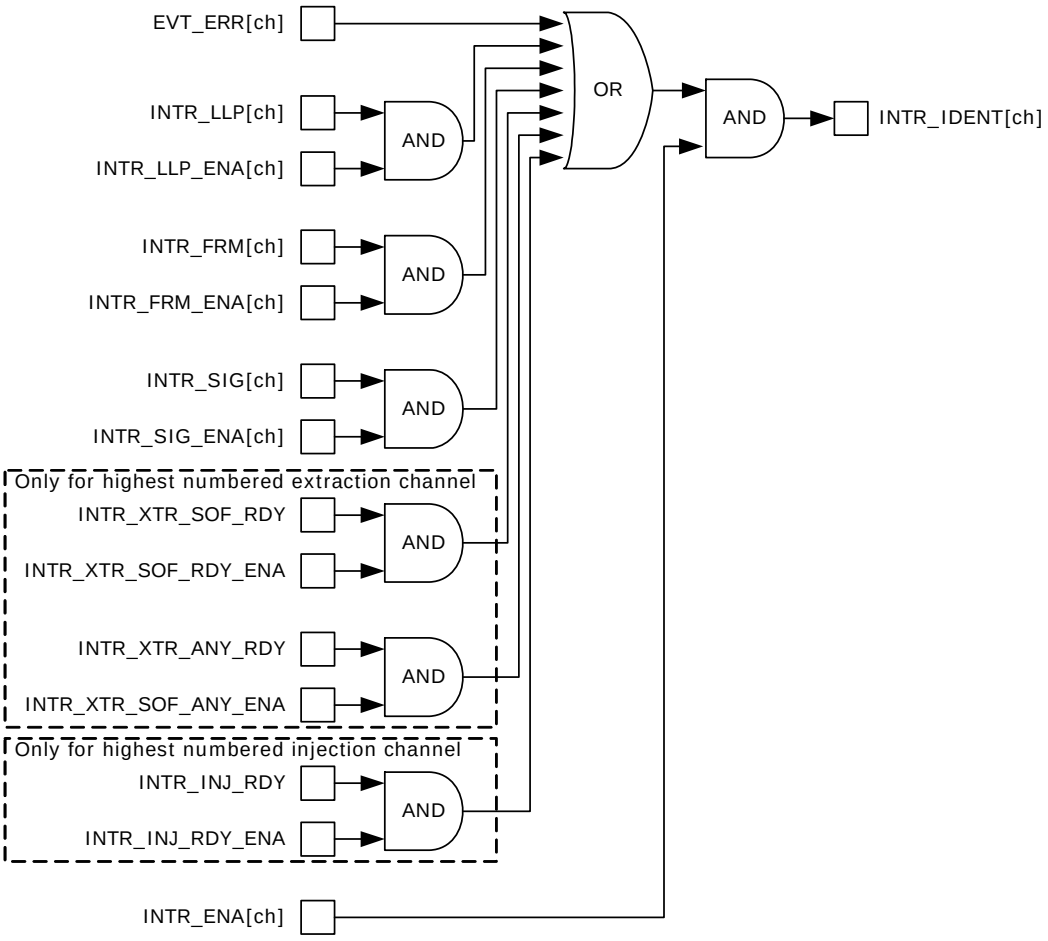
- XTR_SOF_RDY-event is active when the next word to be manually extracted contains an SOF indication. This event is enabled in MANUAL_INTR_ENA.INTR_XTR_SOF_RDY_ENA.
- XTR_ANY_RDY-event is active when any word is available for manually extraction. This event is enabled in MANUAL_INTR_ENA.INTR_XTR_ANY_RDY_ENA respectively.

The highest numbered injection channel supports one additional non-sticky event related to manual injection: INJ_RDY-event. This event is active when the injection logic has room for (at least) sixteen 32-bit words of injection frame data. When this event is active, MANUAL_INTR.INTR_INJ_RDY is set. The INJ_RDY-event can be enabled for channel interrupt by setting MANUAL_INTR_ENA.INTR_INJ_RDY_ENA. For more information, see Manual Injection.

The FDMA_INTR_ENA.INTR_ENA[ch] field enables interrupt from individual channels, FDMA_INTR_IDENT.INTR_IDENT[ch] field shows which channels that are currently interrupting. While INTR_IDENT is non-zero, the FDMA is indicating interrupting towards to the VCore-III interrupt controller.

The following illustration shows the FDMA channel interrupt hierarchy.

Figure 149 • **FDMA Channel Interrupt Hierarchy**



4.7.5 FDMA Extraction

During extraction, the FDMA extracts Ethernet frame data from the Queuing System and saves it into the data block of the DCB that is currently loaded by the FDMA extraction channel. The FDMA continually processes DCBs until it reaches a DCB with LLP = NULL or until it is disabled.

When an extraction channel writes the status word of a DCB, it updates SOF/EOF/ABORT/PRUNED-indications and BLOCKL. BLOCKO remains unchanged (write value is taken from the value that was read when the DCB was loaded).

Aborting frames from the queuing system will not occur during normal operation. If the queuing system is reset during extraction of a particular frame, then ABORT and EOF is set. Software must discard frames that have ABORT set.

Pruning of frames during extraction is enabled per extraction port through XTR_FRM_PRUNING[group].PRUNE_SIZE. When enabled, Ethernet frames above the configured size are truncated, and PD and EOF is set.

4.7.6 FDMA Injection

During injection, the FDMA reads Ethernet frame data from the data block of the DCB that is currently loaded by the FDMA injection channel and injects this into the queuing system. The FDMA continually processes DCBs until it reaches a DCB with LLP = NULL or until it is disabled.

When an injection channel writes the status word of a DCB, it sets DONE indication. All other status word fields remain unchanged (write value is stored the first time the injection channel loads the DCB).

The rate by which the FDMA injects frames can be shaped by using tokens. Each injection channel has an associated token counter (FDMA_CH_INJ_TOKEN_CNT[ich]). A DCB that has the DATAL.TOKEN field set causes the injection channel to deduct one from the token counter before the data block of the DCB can be injected. If the token counter is at 0, the injection channel postpones injection until the channels token counter is set to a value different from 0.

Tokens can be added to the token counter by writing the FDMA_CH_INJ_TOKEN_CNT[ich] register. Tokens can also be added automatically (with a fixed interval) by using the dedicated token tick counter. Setting FDMA_CH_INJ_TOKEN_TICK_RLD[ich] to a value *n* (different from 0) will cause one token to be added to that injection channel every *n* x 200 ns.

4.7.7 Manual Mode

The decision making logic of the FDMA extraction path and/or injection paths can be disabled to give control of the FDMA's extraction and/or injection buffers directly to any master attached to the Shared Bus Architecture. When operating in manual mode DCB structure, counters and most of the interrupts do not apply.

Manual extraction and injection use hard-coded channel numbers.

- Manual extraction mode uses FDMA channel 1 (port 54 extraction direction).
- Manual injection mode uses FDMA channel 9 (port 54 injection direction).

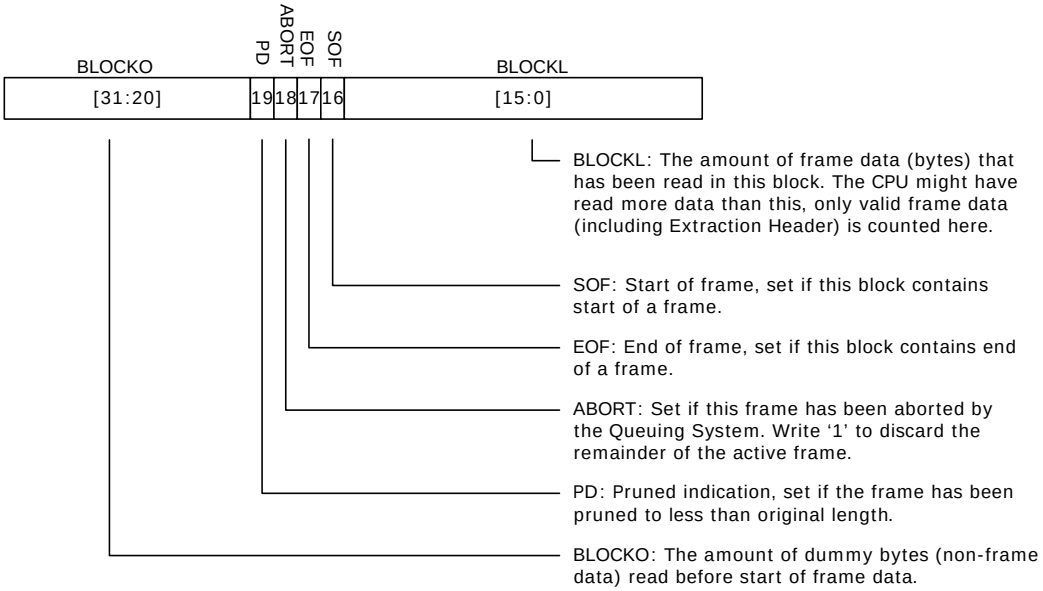
To enable manual extraction, set MANUAL_CFG.XTR_ENA. The FDMA must not be enabled for FDMA extraction when manual extraction mode is active. To enable manual injection, set MANUAL_CFG.INJ_ENA and FDMA_CH_CFG[9].CH_INJ_CRP = 1. The FDMA must not be enabled for FDMA injection when manual injection mode is active. Extraction and injection paths are independent. For example, FDMA can control extraction at the same time as doing manual injection by means of the CPU.

4.7.7.1 Manual Extraction

Extraction is done by reading one or more blocks of data from the extraction memory area. A block of data is defined by reading one or more data words followed by reading of the extraction status word. The extraction memory area is 16 kilobytes and is implemented as a replicated register region MANUAL_XTR. The highest register in the replication (0xFFF) maps to the extraction status word. The status word is updated by the extraction logic and shows the number of frame bytes read, SOF and EOF indications, and PRUNED/ABORT indications.

Note: During frame extraction, the CPU does not know the frame length. This means that the CPU must check for EOF regularly during frame extraction. When reading a data block from the device, the CPU can burst read from the memory area in such a way that the extraction status word is read as the last word of the burst.

Figure 150 **• Extraction Status Word Encoding**



The extraction logic updates the extraction status word while the CPU is reading out data for a block. Prior to starting a new data block, the CPU can write the two least significant bits of the BLOCKO field. The BLOCKO value is stored in the extraction logic and takes effect when the new data block is started. Reading the status field always returns the BLOCKO value that applies to the current data block. Unless written (before stating a data block), the BLOCKO is cleared, so by default all blocks have 0 byte offset. The offset can be written multiple times; the last value takes effect.

The CPU can abort frames that it has already started to read out by writing the extraction status field with the ABORT field set. All other status-word fields will be ignored. This causes the extraction logic to discard the active frame and remove it from the queuing system.

Reading of block data does not have to be done in one burst. As long as the status word is not read, multiple reads from the extraction region are treated as belonging to the same block.

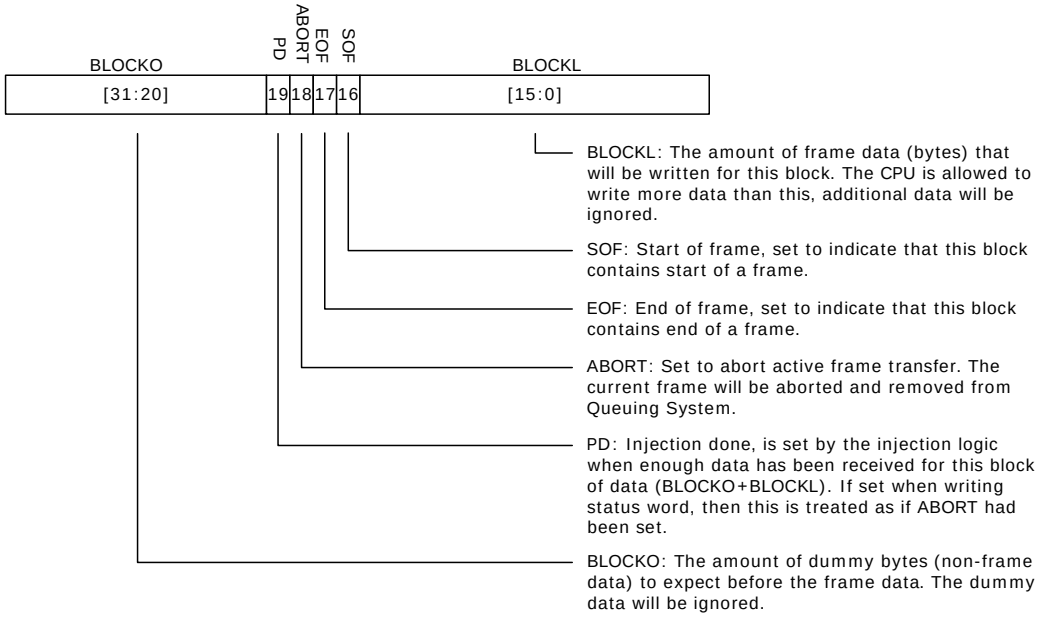
4.7.7.2 Manual Injection

Injection is done by writing one or more blocks of data to the injection memory area. A block of data is defined by writing an injection status word followed by writing one or more data words. The injection memory area is 16 kilobytes and is implemented as a replicated register region MANUAL_INJ. The first register in this replication maps to the injection status word. The status word for each block defines the following:

- Number of bytes to be injected
- Optional byte offset before injection data
- SOF and EOF indications
- Optional ABORT indication

Note: In general, it makes sense to inject frames as a single large block of data (containing both SOF and EOF). However, because offset/length can be specified individually for each block, injecting frames through several blocks is useful when compensating for offset/word misalignment. For example, when building a frame from different memory regions in the CPU main memory.

Figure 151 **• Injection Status Word Encoding**



Injection logic updates the PD field of the status word. The status word can be read at any time during injection without affecting current data block transfers. However, a CPU is able to calculate how much data that is required to inject a complete block of data (at least BLOCKO + BLOCKL bytes), so reading the status word is for software development and debug only. Writing status word before finishing a previous data block will cause that frame to be aborted. Frames are also aborted if they do not start with SOF or end with EOF indications.

As long as the status word is not written, multiple writes to the injection region are treated as data belonging to the same block. This means multiple bursts of data words can be written between each injection status word update.

Software can abort frames by writing to injection status with ABORT field set. All other status word fields are ignored. When a frame is aborted, the already injected data is removed from the queuing system.

4.8 VCore-III System Peripherals

This section describes the subblocks of the VCore-III system. Although the subblocks are primarily intended to be used by the VCore-III CPU, an external CPU can also access and control them through the shared bus.

4.8.1 SI Boot Controller

The SPI boot master allows booting from a Flash that is connected to the serial interface. For information about how to write to the serial interface, see SI Master Controller, page 422. For information about using an external CPU to access device registers using the serial interface, see Serial Interface in Slave Mode, page 399.

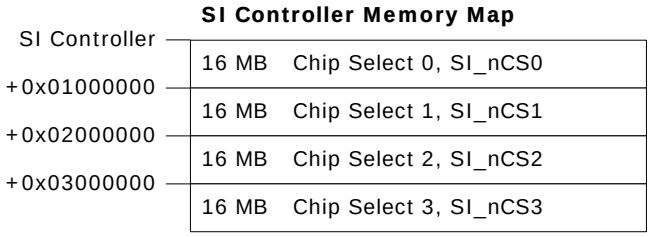
The following table lists the registers associated with the SI boot controller.

Table 247 **• SI Boot Controller Configuration Registers**

Register	Description
ICPU_CFG::SPI_MST_CFG	Serial interface speed and address width
ICPU_CFG::SW_MODE	Legacy manual control of the serial interface pins
ICPU_CFG::GENERAL_CTRL	SI interface ownership

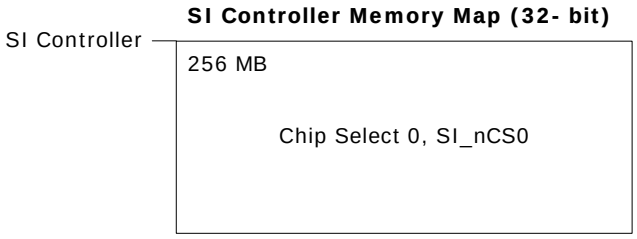
By default, the SI boot controller operates in 24-bit address mode. In this mode, there are four programmable chip selects when the VCore-III system controls the SI. Each chip select can address up to 16 megabytes (MB) of memory.

Figure 152 • **SI Boot Controller Memory Map in 24-Bit Mode**



The SI boot controller can be reconfigured for 32-bit address mode through SPI_MST_CFG.A32B_ENA. In 32-bit mode, the entire SI region of 256 megabytes (MB) is addressed using chip select 0.

Figure 153 • **SI Boot Controller Memory Map in 32-Bit Mode**



Reading from the memory region for a specific SI chip select generates an SI read on that chip select. The VCore-III CPU can execute code directly from Flash by executing from the SI boot controller's memory region. For 32-bit capable SI Flash devices, the VCore-III must start up in 24-bit mode. During the boot process, it must manually reconfigure the Flash device (and SI boot controller) into 32-bit mode and then continue booting.

The SI boot controller accepts 8-bit, 16-bit, and 32-bit read access with or without bursting. Writing to the SI requires manual control of the SI pins using software. Setting SW_MODE.SW_PIN_CTRL_MODE places all SI pins under software control. Output enable and the value of SI_CLK, SI_DO, SI_nCS*n* are controlled through the SW_MODE register. The value of the SI_DI pin is available in SW_MODE.SW_SPI_SDI. The software control mode is provided for legacy reasons; new implementations should use the dedicated master controller for writing to the serial interface. For more information see SI Master Controller, page 422.

Note: The VCore-III CPU cannot execute code directly from the SI boot controller's memory region at the same time as manually writing to the serial interface.

The following table lists the serial interface pins when the SI boot controller is configured as owner of SI interface in GENERAL_CTRL.IF_SI_OWNER. For more information about overlaid functions on the GPIOs for these signals, see GPIO Overlaid Functions, page 439.

Table 248 • **Serial Interface Pins**

Pin Name	I/O	Description
SI_nCS0 SI_nCS[3:1]/GPIO	O	Active low chip selects. Only one chip select can be active at any time. Chip selects 1 through 3 are overlaid functions on GPIO pins.
SI_CLK	O	Clock output.
SI_DO	O	Data output (MOSI).
SI_DI	I	Data input (MISO).

The SI boot controller does speculative prefetching of data. After reading address *n*, the SI boot controller automatically continues reading address *n* + 1, so that the next value is ready if requested by

the VCore-III CPU. This greatly optimizes reading from sequential addresses in the Flash, such as when copying data from Flash into program memory.

The following illustrations depict 24-bit address mode. When the controller is set to 32-bit mode (through SPI_MST_CFG.A32B_ENA), 32 address bits are transferred instead of 24.

Figure 154 • **SI Read Timing in Normal Mode**

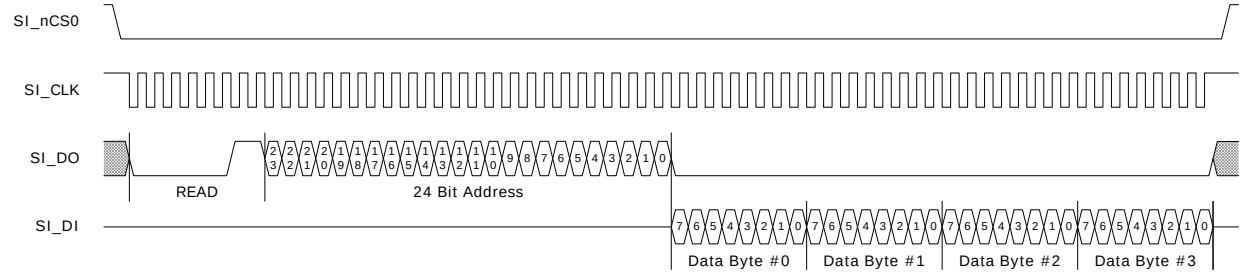
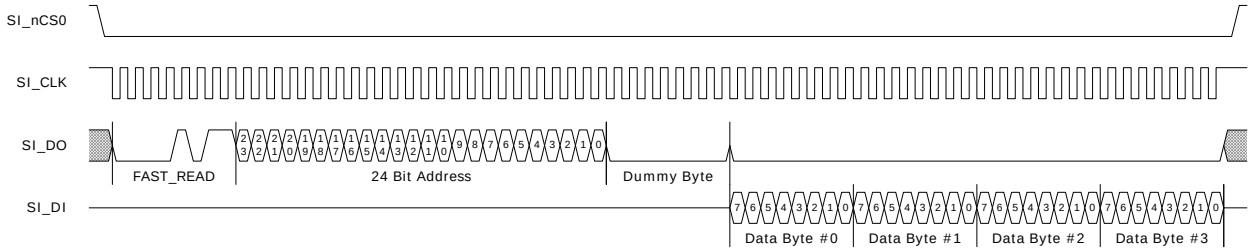


Figure 155 • **SI Read Timing in Fast Mode**



The default timing of the SI boot controller operates with most serial interface Flash devices. Use the following process to calculate the optimized serial interface parameters for a specific SI device:

1. Calculate an appropriate frequency divider value as described in SPI_MST_CFG.CLK_DIV. The SI operates at no more than 25 MHz, and the maximum frequency of the SPI device must not be exceeded. The VCore-III system frequency in the device is 250 MHz.
2. The SPI device may require a FAST_READ command rather than normal READ when the SI frequency is increased. Setting SPI_MST_CFG.FAST_READ_ENA makes the SI boot controller use FAST_READ commands.
3. Calculate SPI_MST_CFG.CS_DESELECT_TIME so that it matches how long the SPI device requires chip-select to be deasserted between access. This value depends on the SI clock period that results from the SPI_MST_CFG.CLK_DIV setting.

These parameters must be written to SPI_MST_CFG. The CLK_DIV field must either be written last or at the same time as the other parameters. The SPI_MST_CFG register can be configured while also booting up from the SI.

When the VCore-III CPU boots from the SI interface, the default values of the SPI_MST_CFG register are used until the SI_MST_CFG is reconfigured with optimized parameters. This implies that SI_CLK is operating at approximately 8.1 MHz, with normal read instructions and maximum gap between chip select operations to the Flash.

Note: The SPI boot master does optimized reading. SI_DI (from the Flash) is sampled just before driving falling edge on SI_CLK (to the Flash). This greatly relaxes the round trip delay requirement for SI_CLK to SI_DI, allowing high Flash clock frequencies.

4.8.2 SI Master Controller

This section describes the SPI master controller (SIMC) and how to use it for accessing external SPI slave devices, such as programming of a serially attached Flash device on the boot interface. VCore booting from serial Flash is handled by the SI boot master.

The following table lists the registers associated with the SI master controller.

Table 249 • **SI Master Controller Configuration Registers Overview**

Register	Description
SIMC::CTRLR0	Transaction configuration
SIMC::CTRLR1	Configurations for receive-only mode
SIMC::SIMCEN	SI master controller enable
SIMC::SER	Slave select configuration
SIMC::BAUDR	Baud rate configuration
SIMC::TXFTLR	Tx FIFO threshold level
SIMC::RXFTLR	Rx FIFO Threshold Level
SIMC::TXFLR	Tx FIFO fill level
SIMC::RXFLR	Rx FIFO fill level
SIMC::SR	Various status indications
SIMC::IMR	Interrupt enable
SIMC::ISR	Interrupt sources
SIMC::RISR	Unmasked interrupt sources
SIMC::TXOICR	Clear of transmit FIFO overflow interrupt
SIMC::RXOICR	Clear of receive FIFO overflow interrupt
SIMC::RXUICR	Clear of receive FIFO underflow interrupt
SIMC::DR	Tx/Rx FIFO access
ICPU_CFG::GENERAL_CTRL	Interface configurations

The SI master controller supports Motorola SPI and Texas Instruments SSP protocols. The default protocol is SPI, enable SSP by setting CTRLR0.FRF = 1 and GENERAL_CTRL.SSP_MODE_ENA = 1. The protocol baud rate is programmable in BAUDR.SCKDV; the maximum baud rate is 25 MHz.

Before the SI master controller can be used, it must be set as owner of the serial interface. This is done by writing GENERAL_CTRL.IF_SI_OWNER = 2.

The SI master controller has a programmable frame size. The frame size is the smallest unit when transferring data over the SI interface. Using CTRLR0.DFS, the frame size is configured in the range of 4 bits to 16 bits. When reading or writing words from the transmit/receive FIFO, the number of bits that is stored per FIFO-word is always equal to frame size (as programmed in CTRLR0.DFS).

The controller operates in one of three following major modes: Transmit and receive, transmit only, or receive only. The mode is configured in CTRLR0.TMOD.

Transmit and receive. software paces SI transactions. For every data frame that software writes to the transmission FIFO, another data frame is made available in the receive FIFO (receive data from the serial interface). Transaction will go on for as long as there is data available in the transmission FIFO.

Transmit only. Software paces SI transactions. The controller transmits only; receive data is discarded. Transaction will go on for as long as there is data available in the transmission FIFO.

Receive only. The controller paces SI transactions. The controller receives only data, software requests a specific number of data frames by means of CTRLR1.NDF, the transaction will go on until all data frames has been read from the SI interface. Transaction is initiated by software writing one data frame to transmission FIFO. This frame is not used for anything else than starting the transaction. The SI_DO output is undefined during receive only transfers. Receive data frames are put into the receive FIFO as they are read from SI.

For SPI mode, chip select is asserted throughout transfers. Transmit transfers end when the transmit FIFO runs out of data frames. Software must make sure it is not interrupted while writing data for a multiframe transfer. When multiple distinct transfers are needed, wait for SR.TFE = 1 and SR.BUSY = 0 before initiating new transfer. SR.BUSY is an indication of ongoing transfers, however, it is not asserted immediately when writing frames to the FIFO. As a result, software must also check the SR.TFE (transmit FIFO empty) indication.

The SI master controller supports up to 16 chip selects. Chip select 0 is mapped to the SI_nCS pin of the serial interface. The remaining chips selects are available using overlaid GPIO functions. Software controls which chip selects to activate for a transfer using the SER register. For more information about overlaid functions on the GPIOs for these signals, see GPIO Overlaid Functions, page 439.

Table 250 • **SI Master Controller Pins**

Pin Name	I/O	Description
SI_nCS0 SI_nCS[15:1]/GPIO	O	Active low chip selects. Chip selects 1 through 15 are overlaid functions on the GPIO pins.
SI_CLK	O	Clock output.
SI_DO	O	Data output (MOSI).
SI_DI	I	Data input (MISO).

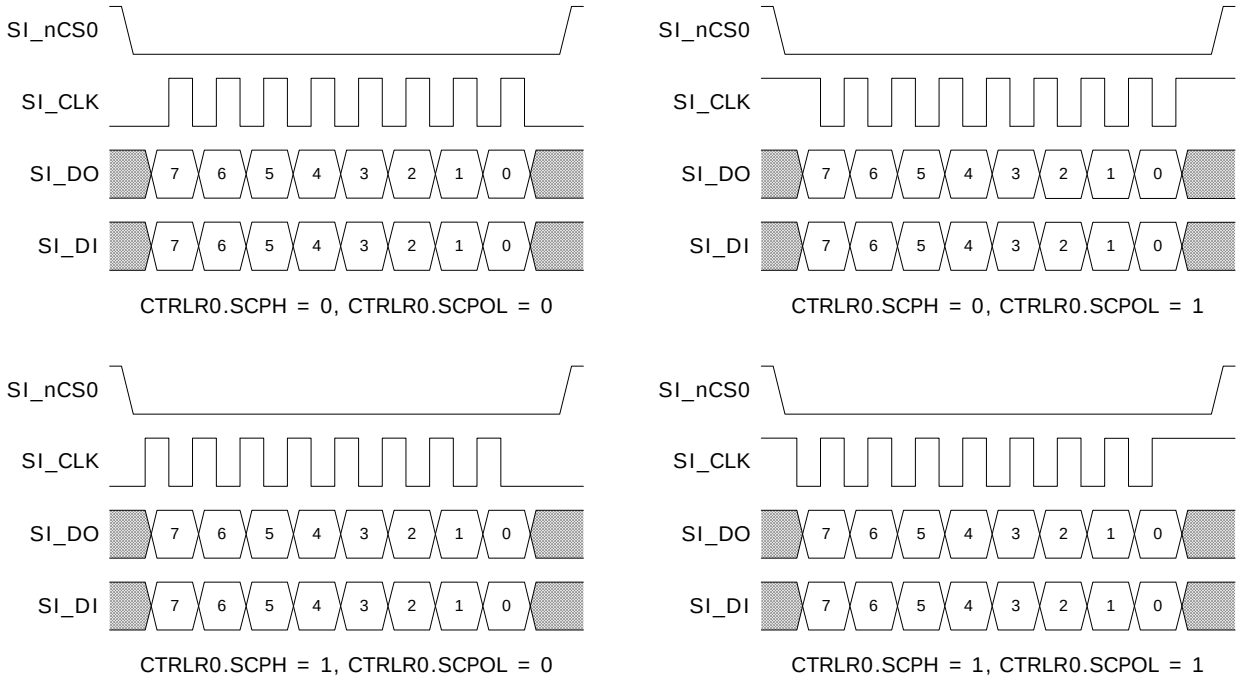
Writing to any DR replication index put data into the transmission FIFO, and reading from any DR replication index take out data from the receive FIFO. FIFO status indications are available in the SR register's TFE, TFNF, RFF, and RFNE fields. For information about interrupting on FIFO status, see SIMC Interrupts, page 426.

After completing all initial configurations, the SI master controller is enabled by writing SIMCEN.SIMCEN = 1. Some registers can only be written when the controller is in disabled state. This is noted in the register-descriptions for the affected registers.

4.8.2.1 SPI Protocol

When the SI master controller is configured for SPI mode, clock polarity and position is configurable in CTRLR0.SCPH and CTRLR0.SCPOL. The following illustration shows the four possible combinations for 8-bit transfers.

Figure 156 **• SIMC SPI Clock Configurations**

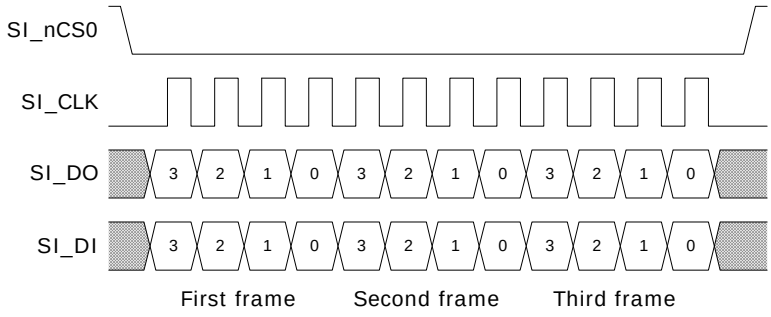


Data is sampled in the middle of the data-eye.

When using SPI protocol and transferring more than one data frame at the time, the controller performs one consecutive transfer. The following illustration shows a transfer of three data frames of 4 bits each.

Note: **T**ransmitting transfers end when the transmit FIFO runs out of data frames. Receive only transfers end when the pre-defined number of data-frames is read from the SI interface.

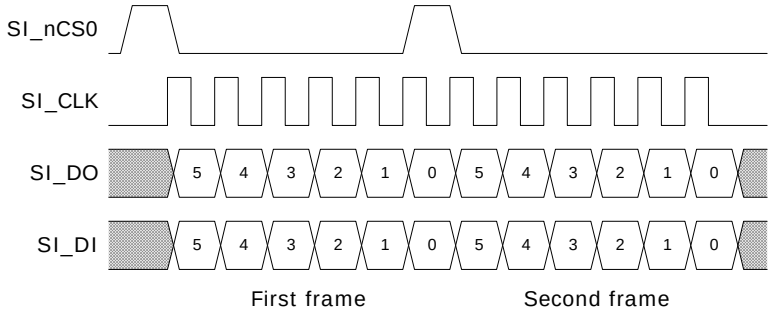
Figure 157 **• SIMC SPI 3x Transfers**



4.8.2.2 SSP Protocol

The SSP protocol for transferring two 6-bit data frames is shown in the following illustration. When using SSP mode, CTRLR0.SCPH and CTRLR0.SCPOL must remain zero.

4.8.2.3 SIMC SSP 2x Transfers



4.8.2.4 SIMC Interrupts

The SI master controller has five interrupt sources:

- RXF interrupt is asserted when the fill level of the receive FIFO (available in RXFLR) exceeds the programmable level set in RXFTLR. This interrupt is non-sticky; it is deasserted when fill level is decreased below the programmable level.
- RXO interrupt is asserted on receive FIFO overflow. This interrupt is cleared by reading the RXOICR register.
- RXU interrupt is asserted when reading from an empty receive FIFO. This interrupt is cleared by reading the RXUICR register.
- TXO interrupt is asserted when writing to a full transmit FIFO. This interrupt is cleared by reading the TXOICR register.
- TXE interrupt is asserted when the fill level of the transmit FIFO (available in TXFLR) is below the programmable level set in TXFTLR. This interrupt is non-sticky; it is deasserted when fill level is increased above the programmable level.

The raw (unmasked) status of these interrupts is available in the RISR register. Each of the interrupts can be individually masked by the IMR register. All interrupts are enabled by default. The currently active interrupts are shown in the ISR register.

If the RXO, RXU, and TXO interrupts occur during normal use of the SI master controller, it is an indication of software problems.

Interrupt is asserted towards the VCore-III interrupt controller when any enabled interrupt is asserted and the SI master controller is enabled.

4.8.2.5 SIMC Programming Example

This example shows how to use the SI master controller to interface with the SI slave of another device. The slave device will be initialized for big endian/most significant bit first mode and then the DEVCPU_GCB::GPR register is written with the value 0x01234567. It is assumed that the other device's SI slave is connected on SI_nCS1 (and that appropriate GPIO alternate function has been enabled).

The SI slave is using a 24-bit address field and a 32-bit data field. This example uses 4 x 16-bit data frames to transfer the write-access. This means that 8 bits too much will be transferred, this is not a problem for the Microsemi SI slave interface; it ignores data above and beyond the 56 bit required for a write-access.

For information about initialization and how to calculate the 24-bit SI address field, see Serial Interface in Slave Mode, page 399. The address-field when writing DEVCPU_GCB::GPR is 0x804001 (including write-indication).

The following steps are required to bring up the SI master controller and write twice to the SI slave device.

Note: The following procedure disconnects the SI boot master from the SI interface. Booting must be done before attempting to overtake the boot-interface.

1. Write GENERAL_CTRL.IF_SI_OWNER = 2 to make SI master controller the owner of the SI interface.
2. Write BAUDR = 250 to get 1 MHz baud rate.
3. Write SER = 2 to use SI_nCS1.
4. Write CTRLR0 = 0x10F for 16-bit data frame and transmit only.
5. Write SIMCEN = 1 to enable the SI master controller.
6. Write DR[0] = 0x8000, 0x0081, 0x8181, 0x8100 to configure SI slave for big endian / most significant bit first mode.
7. Wait for SR.TFE = 1 and SR.BUSY = 0 for chip select to deassert between accesses to the SI slave.
8. Write DR[0] = 0x8040, 0x0101, 0x2345, 0x6700 to write DEVCPU_GCB::GPR in slave device.
9. Wait for SR.TFE = 1 and SR.BUSY = 0, then disable the SI master controller by writing SIMCEN = 0.

When reading from the SI slave, CTRLR0.TMOD must be configured for transmit and receive. One-byte padding is appropriate for a 1 MHz baud rate.

4.8.3 DDR3/DDR3L Memory Controller

This section provides information about how to configure and initialize the DDR3/DDR3L memory controller.

The following table lists the registers associated with the memory controller.

Table 251 • **DDR3/DDR3L Controller Registers**

Register	Description
ICPU_CFG::RESET	Reset Control
ICPU_CFG::MEMCTRL_CTRL	Control
ICPU_CFG::MEMCTRL_CFG	Configuration
ICPU_CFG::MEMCTRL_STAT	Status
ICPU_CFG::MEMCTRL_REF_PERIOD	Refresh period configuration
ICPU_CFG::MEMCTRL_ZQCAL	DDR3/DDR3L ZQ-calibration
ICPU_CFG::MEMCTRL_TIMING0	Timing configuration
ICPU_CFG::MEMCTRL_TIMING1	Timing configuration
ICPU_CFG::MEMCTRL_TIMING2	Timing configuration
ICPU_CFG::MEMCTRL_TIMING3	Timing configuration
ICPU_CFG::MEMCTRL_MR0_VAL	Mode register 0 value
ICPU_CFG::MEMCTRL_MR1_VAL	Mode register 1 value
ICPU_CFG::MEMCTRL_MR2_VAL	Mode register 2 value
ICPU_CFG::MEMCTRL_MR3_VAL	Mode register 3 value
ICPU_CFG::MEMCTRL_TERMRES_CTRL	Termination resistor control
ICPU_CFG::MEMCTRL_DQS_DLY	DQS window configuration
ICPU_CFG::MEMPHY_CFG	SSTL configuration
ICPU_CFG::MEMPHY_ZCAL	SSTL calibration

The memory controller works with JEDEC-compliant DDR3/DDR3L memory modules. The controller runs 312.50 MHz and supports one or two byte lanes in either single or dual row configurations (one or two chip selects). The controller supports up to 16 address bits. The maximum amount of physical memory that can be attached to the controller is 1 gigabyte.

The memory controller supports ECC encoding/decoding by using one of the two lanes for ECC information. Enabling ECC requires that both byte lanes are populated. When ECC is enabled, half the total physical memory is available for software use; the other half is used for ECC information.

The following steps are required to bring up the memory controller.

1. Release the controller from reset by clearing RESET.MEM_RST_FORCE.
2. Configure timing and mode parameters. These depend on the DDR3/DDR3L module and configuration selected for the device. For more information, see DDR3/DDR3L Timing and Mode Configuration Parameters, page 428.
3. Enable and calibrate the SSTL I/Os. For more information, see Enabling and Calibrating SSTL I/Os.
4. Initialize the memory controller and modules. For more information, see Memory Controller and Module Initialization, page 430.
5. Calibrate the DQS read window. For more information, see DQS Read Window Calibration, page 431.
6. Optionally configure ECC and 1 gigabyte support. For more information, see ECC and 1-Gigabyte Support, page 432.

4.8.3.1 DDR3/DDR3L Timing and Mode Configuration Parameters

This section provides information about each of the parameters that must be configured prior to initialization of the memory controller. It provides a quick overview of fields that must be configured and their recommended values. For more information about each field, see the register list.

The memory controller operates at 312.5 MHz, and the corresponding DDR clock period is 3.2 ns (DDR625). When a specific timing value is used in this section (for example, t_{MOD}), it means that the corresponding value from the memory module datasheet is expressed in memory controller clock cycles (divided by 3.2 ns and rounded up and possibly adjusted for minimum clock cycle count).

The following table defines general parameters for DDR3/DDR3L required when configuring the memory controller.

Table 252 • **General DDR3/DDR3L Timing and Mode Parameters**

Parameter	DDR3/DDR3L
RL	CL
WL	CWL
MD	t_{MOD}
ID	t_{XPR}
SD	t_{DLLK}
OW	2
OR	2
RP	t_{RP}
FAW	t_{FAW}
BL	4
MR0	((RL-4)<<4) | ((t_{WR-4})<<9)
MR1	0x0040
MR2	((WL-5)<<3)
MR3	0x0000

The following table lists the memory controller registers that must be configured before initialization. It is recommended to configure all fields listed, even when ODT or dual row configurations are not used.

Table 253 • **Memory Controller Configuration**

Register Field	Description
MEMCTRL_CFG.DDR_MODE	Set to 1 for DDR3/DDR3L mode.

Table 253 • **Memory Controller Configuration *(continued)***

Register Field	Description
MEMCTRL_CFG.DDR_WIDTH	Set to 0 when using 1 byte lane. Set to 1 when using 2 byte lanes.
MEMCTRL_CFG.MSB_COL_ADDR	Set to one less than the number of column address bits for the modules.
MEMCTRL_CFG.MSB_ROW_ADDR	Set to one less than the number of row address bits for the modules. See field description for example.
MEMCTRL_CFG.BANK_CNT	Set to 1.
MEMCTRL_CFG.BURST_LEN	Set to 1.
MEMCTRL_CFG.BURST_SIZE	Set to same value as MEMCTRL_CFG.DDR_WIDTH.
MEMCTRL_CFG.DDR_ECC_ENA	Set to 0 during initialization of the memory controller.
MEMCTRL_CFG.DDR_ECC_COR_ENA	Set to 0 during initialization of the memory controller.
MEMCTRL_CFG.DDR_ECC_ERR_ENA	Set to 0 during initialization of the memory controller.
MEMCTRL_CFG.DDR_512MBYTE_PLUS	Set to 0 during initialization of the memory controller.
MEMCTRL_REF_PERIOD.REF_PERIOD	Set to (tREFI(ns)/3.2ns) round down to the nearest integer.
MEMCTRL_REF_PERIOD.MAX_PEND_REF	Set to 8.
MEMCTRL_TIMING0.RD_DATA_XFR_DLY	Set to (RL-3).
MEMCTRL_TIMING0.WR_DATA_XFR_DLY	Set to (WL-1).
MEMCTRL_TIMING0.RD_TO_PRECH_DLY	Set to (BL-1).
MEMCTRL_TIMING0.WR_TO_PRECH_DLY	Set to (WL+BL+tWR-1).
MEMCTRL_TIMING0.RAS_TO_PRECH_DLY	Set to (tRAS(min)-1).
MEMCTRL_TIMING0.RD_CS_CHANGE_DLY	Set to BL.
MEMCTRL_TIMING0.WR_CS_CHANGE_DLY	Set to (BL-1).
MEMCTRL_TIMING0.RD_TO_WR_DLY	Set to (RL+BL+1-WL).
MEMCTRL_TIMING1.WR_TO_RD_DLY	Set to (WL+BL+tWTR-1).
MEMCTRL_TIMING1.RAS_TO_CAS_DLY	Set to (tRCD-1).
MEMCTRL_TIMING1.RAS_TO_RAS_DLY	Set to (tRRD-1).
MEMCTRL_TIMING1.PRECH_TO_RAS_DLY	Set to (tRP-1).
MEMCTRL_TIMING1.BANK8_FAW_DLY	Set to (FAW-1).
MEMCTRL_TIMING1.RAS_TO_RAS_SAME_BANK_DLY	Set to (tRC-1).
MEMCTRL_TIMING2.INIT_DLY	Set to (ID-1) during initialization of the memory controller.
MEMCTRL_TIMING2.REF_DLY	Set to (tRFC-1).
MEMCTRL_TIMING2.MDSET_DLY	Set to (MD-1).
MEMCTRL_TIMING2.PRECH_ALL_DLY	Set to (RP-1).
MEMCTRL_TIMING3.WR_TO_RD_CS_CHANGE_DLY	Set to (WL+tWTR-1).
MEMCTRL_TIMING3.LOCAL_ODT_RD_DLY	Set to (RL-3).
MEMCTRL_TIMING3.ODT_WR_DLY	Set to (OW-1).
MEMCTRL_TIMING3.ODT_RD_DLY	Set to (OR-1).
MEMCTRL_MR0_VAL.MR0_VAL	Set to MR0.

Table 253 • **Memory Controller Configuration *(continued)***

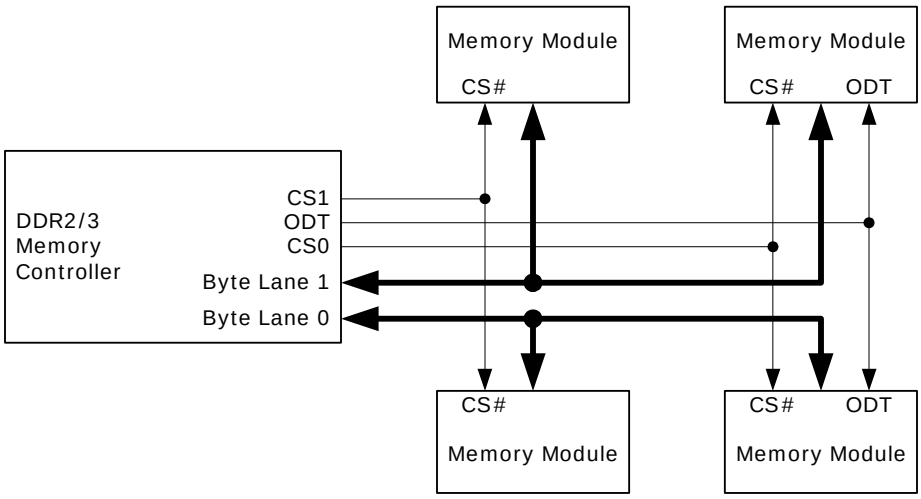
Register Field	Description
MEMCTRL_MR1_VAL.MR1_VAL	Set to MR1.
MEMCTRL_MR2_VAL.MR2_VAL	Set to MR2.
MEMCTRL_MR3_VAL.MR3_VAL	Set to MR3.

The memory controller supports two chip selects and automatically selects modules based on the configured address width and bank count. The resulting pool of memory is one large consecutive region.

4.8.3.2 On-Die Termination

An end termination scheme must be configured when writing from the device to the memory modules, the memory module furthest away from the device must be connected to the ODT output. Enable assertion of ODT output for all write access by setting MEMCTRL_TERMRES_CTRL = 0x0000000C.

Figure 158 • **Memory Controller ODT Hookup**



4.8.3.3 Enabling and Calibrating SSTL I/Os

Prior to controller initialization, the device's SSTL I/O drivers must be enabled and calibrated to correct drive strength and termination resistor values. It is recommended that the device's I/O drive and termination strengths be 60 Ω/60 Ω.

Complete the following tasks to enable and calibrate the SSTL I/Os.

1. Enable the SSTL mode by clearing MEMPHY_CFG.PHY_RST and setting MEMPHY_CFG.PHY_SSTL_ENA.
2. Perform calibration with the previously mentioned drive and termination strengths by writing 0xEF to MEMPHY_ZCAL.
3. Wait until MEMPHY_ZCAL.ZCAL_ENA is cleared (indicates calibration is done).
4. Enable drive of the SSTL I/Os by setting MEMPHY_CFG.PHY_CK_OE, MEMPHY_CFG.PHY_CL_OE, and MEMPHY_CFG.PHY_ODT_OE.

The SSTL interface is now enabled and calibrated, and the initialization of the memory controller can commence.

4.8.3.4 Memory Controller and Module Initialization

After memory controller parameters are configured, and after the SSTL I/Os are enabled and calibrated, the memory controller and modules are initialized by setting MEMCTRL_CTRL.INITIALIZE.

During initialization, the memory controller automatically follows the proper JEDEC defined procedure for initialization and writing of mode registers to the DDR3/DDR3L memory modules.

The memory controller automatically clears MEMCTRL_CTRL.INITIALIZE and sets the MEMCTRL_STAT.INIT_DONE field after the controller and the memory module (or modules) are operational. Software must wait for the INIT_DONE indication before continuing to calibrate the read window.

4.8.3.5 DQS Read Window Calibration

After initialization of the memory controller, successful writes to the memory are guaranteed. Reading is not yet possible, because the round trip delay between controller and memory modules are not calibrated.

Calibration of the read window includes writing a known value to the start of the DDR memory and then continually reading this value while adjusting the DQS window until the correct value is read from the memory. The calibration procedure depends on the number of byte lanes that are used (see MEMCTRL_CFG.DDR_WIDTH).

Note: For an ECC system, the calibration procedure for two byte lanes must be used, because the memory controller is initialized without ECC enabled.

4.8.3.5.1 Calibration of One Byte Lane

The following procedure applies to systems where MEMCTRL_CFG.DDR_WIDTH is 0:

- Write 0x000000FF to SBA address 0x20000000.
- Set the MEMCTRL_DQS_DLY[0].DQS_DLY field to 0.

Perform the following steps to calibrate the read window. Do not increment the DQS_DLY field beyond its maximum value. If the DQS_DLY maximum value is exceeded, it is an indication something is incorrect, and the memory controller will not be operational.

1. Read byte address 0x20000000. If the content is different from 0xFF, increment MEMCTRL_DQS_DLY[0].DQS_DLY by one, and repeat step 1, else continue to step 2.
2. Read byte address 0x20000000. If the content is different from 0x00, increment MEMCTRL_DQS_DLY[0].DQS_DLY by one, and repeat step 2, else continue to step 3.
3. Decrement MEMCTRL_DQS_DLY[0].DQS_DLY by three.

After the last step, which configures the appropriate DQS read window, the memory is ready for random read and write operations. For more information about ECC or 1 gigabyte support, see ECC and 1-Gigabyte Support, page 432.

4.8.3.5.2 Calibration of Two Byte Lanes

This procedure applies to systems where MEMCTRL_CFG.DDR_WIDTH is 1:

- Write 0x0000FFFF to SBA address 0x20000000.
- Write 0x00000000 to SBA address 0x20000004.
- Set the MEMCTRL_DQS_DLY[0].DQS_DLY field to 0.
- Set the MEMCTRL_DQS_DLY[1].DQS_DLY field to 0.

Perform the following steps to calibrate the read window. Do not increment the DQS_DLY fields beyond their maximum value. If a DQS_DLY maximum value is exceeded, it is an indication something is incorrect, and the memory controller will not be operational.

1. Read byte addresses 0x20000000 and 0x20000001 from memory.
   If the contents of byte address 0x20000000 is different from 0xFF, increment MEMCTRL_DQS_DLY[0].DQS_DLY by one.

   If the contents of byte address 0x20000001 is different from 0xFF, increment MEMCTRL_DQS_DLY[1].DQS_DLY by one.

   If any DQS_DLY was incremented repeat step 1, else continue to step 2.
2. Read byte addresses 0x20000000 and 0x20000001 from memory.
   If the contents of byte address 0x20000000 is different from 0x00, increment MEMCTRL_DQS_DLY[0].DQS_DLY by one.

   If the contents of byte address 0x20000001 is different from 0x00, increment

MEMCTRL_DQS_DLY[1].DQS_DLY by one.

If any DQS_DLY was incremented repeat step 2, else continue to step 3.

3. Decrement MEMCTRL_DQS_DLY[0].DQS_DLY by three, and decrement MEMCTRL_DQS_DLY[1] by three.

The last step configures the appropriate DQS read windows. After this step the memory is ready for random read and write operations. If ECC or 1 gigabyte is needed, see ECC and 1 Gigabyte Support.

4.8.3.6 ECC and 1-Gigabyte Support

The memory controller supports ECC, allowing correction of single bit and detection of two bit errors in every byte that is read from memory.

When ECC is enabled, one byte lane is used for data words and the other is used for ECC check-word. After enabling ECC, the amount of memory available to software is half of the physical memory (the lower half of the memory region).

ECC support must be enabled just after read calibration by setting MEMCTRL_CFG.DDR_ECC_ENA = 1 and at the same time setting MEMCTRL_CFG.BURST_SIZE = 0.

When ECC is enabled, the memory controller automatically corrects single-bit errors and detects two-bit errors. It is possible to propagate these events to the SBA to trigger a bus-error event in the VCore-III CPU. Non-correctable error indications are forwarded when MEMCTRL_CFG.DDR_ECC_ERR_ENA = 1. Correctable “errors” are forwarded when MEMCTRL_CFG.DDR_ECC_COR_ENA = 1. The VCore-III CPU can generate interrupt on these events so that software can take the appropriate action.

The memory controller supports up to 1 gigabyte of physical memory. To access more than 512 megabytes from the VCore-III CPU, set MEMCTRL_CFG.DDR_512MBYTE_PLUS = 1 to change the SBA memory map. For more information, see Shared Bus. When ECC is enabled with 1 gigabyte of physical memory, MEMCTRL_CFG.DDR_512MBYTE_PLUS does not have to be set, because only half of the physical memory is available to software.

4.8.4 Timers

This section provides information about the timers. The following table lists the registers associated with timers.

Table 254 • **Timer Registers**

Register	Description
ICPU_CFG::TIMER_CTRL	Enable/disable timer
ICPU_CFG::TIMER_VALUE	Current timer value
ICPU_CFG::TIMER_RELOAD_VALUE	Value to load when wrapping
ICPU_CFG::TIMER_TICK_DIV	Common timer-tick divider

There are three decrementing 32-bit timers in the VCore-III system that run from a common divider. The common divider is driven by a fixed 250 MHz clock and can generate timer ticks from 0.1 µs (10 MHz) to 1 ms (1 kHz), configurable through TIMER_TICK_DIV. The default timer tick is 100 µs (10 kHz).

Software can access each timer value through the TIMER_VALUE[*n*] registers. These can be read or written at any time, even when the timers are active.

When a timer is enabled through TIMER_CTRL[*n*].TIMER_ENA, it decrements from the current value until it reaches zero. An attempt to decrement a TIMER_VALUE[*n*] of zero generates interrupt and assigns TIMER_VALUE[*n*] to the contents of TIMER_RELOAD_VALUE[*n*]. Interrupts generated by the timers are sent to the VCore-III interrupt controller. From here, interrupts can be forwarded to the VCore-III CPU or to an external CPU. For more information, see Interrupt Controller.

By setting TIMER_CTRL[*n*].ONE_SHOT_ENA, the timer disables itself after generating one interrupt. By default, timers will decrement, interrupt, and reload indefinitely (or until disabled by clearing TIMER_CTRL[*n*].TIMER_ENA).

A timer can be reloaded from TIMER_RELOAD_VALUE[*n*] at the same time as it is enabled by setting TIMER_CTRL[*n*].FORCE_RELOAD and TIMER_CTRL[*n*].TIMER_ENA at the same time.

For example, configure Timer0 to interrupt every 1 ms. With the default timer tick of 100 µs ten timer ticks are needed for a timer that wraps every 1 ms. Configure TIMER_RELOAD_VALUE[0] to 0x9, then enable the timer and force a reload by setting TIMER_CTRL[0].TIMER_ENA and TIMER_CTRL[0].FORCE_RELOAD at the same time.

4.8.5 UARTs

This section provides information about the UART (Universal Asynchronous Receiver/Transmitter) controllers. There are two independent UART controller instances in the VCore-III system: UART and UART2. These instances are identical copies and anywhere in this description the word UART can be replaced by UART2.

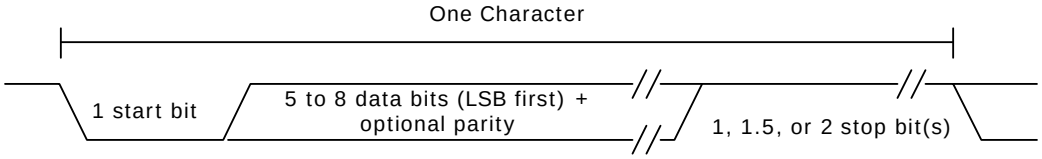
The following table lists the registers associated with the UART.

Table 255 • **UART Registers**

Register	Description
UART::RBR_THR	Receive buffer/transmit buffer/Divisor (low)
UART::IER	Interrupt enable/divisor (high)
UART::IIR_FCR	Interrupt identification/FIFO control
UART::LCR	Line control
UART::MCR	Modem control
UART::LSR	Line status
UART::MSR	Modem status
UART::SCR	Scratchpad
UART::USR	UART status

The VCore-III system UART is functionally based on the industry-standard 16550 UART (RS232 protocol). This implementation features a 16-byte receive and a 16-byte transmit FIFO.

Figure 159 • **UART Timing**



The number of data bits, parity, parity-polarity, and stop-bit lengths are all programmable using LCR.

The UART pins on the device are overlaid functions on the GPIO interface. Before enabling the UART, the VCore-III CPU must enable overlaid modes for the appropriate GPIO pins. For more information about overlaid functions on the GPIOs for these signals, see GPIO Overlaid Functions, page 439.

The following table lists the pins of the UART interfaces.

Table 256 • **UART Interface Pins**

Pin Name	I/O	Description
UART_RXD/GPIO	I	UART receive data

Table 256 • **UART Interface Pins *(continued)***

Pin Name	I/O	Description
UART_TXD/GPIO	O	UART transmit data
UART2_RXD/GPIO	I	UART2 receive data
UART2_TXD/GPIO	O	UART2 transmit data

The baud rate of the UART is derived from the VCore-III system frequency. The divider value is indirectly set through the RBR_THR and IER registers. The baud rate is equal to the VCore-III system clock frequency, divided by 16, and multiplied by the value of the baud rate divisor. A divider of zero disables the baud rate generator and no serial communications occur. The default value for the divisor register is zero.

For example, configure a baud rate of 9600 in a 250 MHz VCore-III system. To generate a baud rate of 9600, the divisor register must be set to 0x65C (250 MHz/(16 × 9600 Hz)). Set LCR.DLAB and write 0x5C to RBR_THR and 0x06 to IER (this assumes that the UART is not in use). Finally, clear LCR.DLAB to change the RBR_THR and IER registers back to the normal mode.

By default, the FIFO mode of the UART is disabled. Enabling the 16-byte receive and 16-byte transmit FIFOs (through IIR_FCR) is recommended.

Note: Although the UART itself supports RTS and CTS, these signals are not available on the pins of the device.

4.8.5.1 UART Interrupt

The UART can generate interrupt whenever any of the following prioritized events are enabled (through IER).

- Receiver error
- Receiver data available
- Character timeout (in FIFO mode only)
- Transmit FIFO empty or at or below threshold (in programmable THRE interrupt mode)

When an interrupt occurs, the IIR_FCR register can be accessed to determine the source of the interrupt. Note that the IIR_FCR register has different purposes when reading or writing. When reading, the interrupt status is available in bits 0 through 3. For more information about interrupts and how to handle them, see the IIR_FCR register description.

For example, enable interrupt when transmit fifo is below one-quarter full. To get this type of interrupt, the THRE interrupt must be used. First, configure TX FIFO interrupt level to one-quarter full by setting IIR_FCR.TET to 10; at the same time, ensure that the IIR_FCR.FIFOE field is also set. Set IER.PTIME to enable the THRE interrupt in the UART. In addition, the VCore-III interrupt controller must be configured for the CPU to be interrupted. For more information, see Interrupt Controller.

4.8.6 Two-Wire Serial Interface

This section provides information about the two-wire serial interface controllers. There are two independent two-wire serial interface controller instances in the VCore-III system, TWI and TWI2. These instances are identical copies and anywhere in this description the word TWI can be replaced by TWI2.

The following table lists the registers associated with the two-wire serial interface.

Table 257 • **Two-Wire Serial Interface Registers**

Register	Description
TWI::CFG	General configuration.
TWI::TAR	Target address.
TWI::SAR	Slave address.
TWI::DATA_CMD	Receive/transmit buffer and command.

Table 257 • **Two-Wire Serial Interface Registers *(continued)***

Register	Description
TWI::SS_SCL_HCNT	Standard speed high time clock divider.
TWI::SS_SCL_LCNT	Standard speed low time clock divider.
TWI::FS_SCL_HCNT	Fast speed high time clock divider.
TWI::FS_SCL_LCNT	Fast speed low time clock divider.
TWI::INTR_STAT	Masked interrupt status.
TWI::INTR_MASK	Interrupt mask register.
TWI::RAW_INTR_STAT	Unmasked interrupt status.
TWI::RX_TL	Receive FIFO threshold for RX_FULL interrupt.
TWI::TX_TL	Transmit FIFO threshold for TX_EMPTY interrupt.
TWI::CLR_*	Individual CLR_* registers are used for clearing specific interrupts. See register descriptions for corresponding interrupts.
TWI::CTRL	Control register.
TWI::STAT	Status register.
TWI::TXFLR	Current transmit FIFO level.
TWI::RXFLR	Current receive FIFO level.
TWI::TX_ABRT_SOURCE	Arbitration sources.
TWI::SDA_SETUP	Data delay clock divider.
TWI::ACK_GEN_CALL	Acknowledge of general call.
TWI::ENABLE_STATUS	General two-wire serial controller status.
ICPU_CFG::TWI_CONFIG	Configuration of TWI_SDA hold-delay.
ICPU_CFG::TWI_SPIKE_FILTER_CFG	Configuration of TWI_SDA spike filter.

The two-wire serial interface controller is compatible with the industry standard two-wire serial interface protocol. The controller supports standard speed up to 100 kbps and fast speed up to 400 kbps. Multiple bus masters, as well as both 7-bit and 10-bit addressing, are also supported.

The two-wire serial interface controller operates as master only.

The two-wire serial interface pins on the device are overlaid functions on the GPIO interface. Before enabling the two-wire serial interface, the VCore-III CPU must enable overlaid functions for the appropriate GPIO pins. For more information about overlaid functions on the GPIOs for these signals, see GPIO Overlaid Functions, page 439.

The following table lists the pins of the two-wire serial interface.

Table 258 • **Two-Wire Serial Interface Pins**

Pin Name	I/O	Description
TWI_SCL/GPIO	I/O	Two-wire serial interface clock, open-collector output.
TWI_SDA/GPIO	I/O	Two-wire serial interfacedata, open-collector output.
TWI_SCL_M*n*/GPIO	I/O	Two-wire serial interface multiplexed clocks (12 instances in total), open-collector outputs.
TWI_SCL_M*n*_AD/GPIO	I/O	Two-wire serial interface multiplexed clocks (4 instances in total). Same as TWI_SCL_M*n*/GPIO but the pin is always driven meaning feedback from slave devices is not possible.
TWI2_SCL/GPIO	I/O	Two-wire serial interface 2 clock, open-collector output.

Table 258 • **Two-Wire Serial Interface Pins *(continued)***

Pin Name	I/O	Description
TWI2_SDA/GPIO	I/O	Two-wire serial interface 2 data, open-collector output.

Setting CTRL.ENABLE enables the controller. The controller can be disabled by clearing the CTRL.ENABLE field, there is a chance that disabling is not allowed (at the time when it is attempted); the ENABLE_STATUS register shows if the controller was successful disabled.

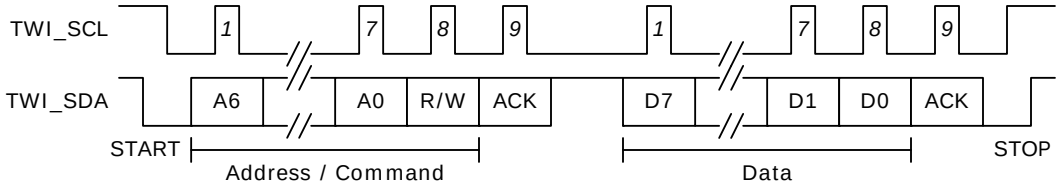
Before enabling the controller, the user must decide on either standard or fast mode (CFG.SPEED) and configure clock dividers for generating the correct timing (SS_SCL_HCNT, SS_SCL_LCNT, FS_SCL_HCNT, FS_SCL_LCNT, and SDA_SETUP). The configuration of the divider registers depends on the VCore-III system clock frequency. The register descriptions explain how to calculate the required values.

Some two-wire serial devices require a hold time on TWI_SDA after TWI_SCL when transmitting from the two-wire serial interface controller. The device supports a configurable hold delay through the TWI_CONFIG register.

The two-wire serial interface controller has an 8-byte combined receive and transmit FIFO.

During normal operation of the two-wire serial interface controller, the STATUS register shows the activity and FIFO states.

Figure 160 • **Two-Wire Serial Interface Timing for 7-bit Address Access**



4.8.6.1 Two-Wire Serial Interface Addressing

To configure either 7-bit or 10 bit addressing, use CFG.MASTER_10BITADDR.

The two-wire serial interface controller can generate both General Call and START Byte. Initiate this through TAR.GC_OR_START_ENA or TAR.GC_OR_START. The target/slave address is configured using the TAR register.

4.8.6.2 Two-Wire Serial Interface Interrupt

The two-wire serial interface controller can generate a multitude of interrupts. All of these are described in the RAW_INTR_STAT register. The RAW_INTR_STAT register contains interrupt fields that are always set when their trigger conditions occur. The INTR_MASK register is used for masking interrupts and allowing interrupts to propagate to the INTR_STAT register. When set in the INTR_STAT register, the two-wire serial interface controller asserts interrupt toward the VCore-III interrupt controller.

The RAW_INTR_STAT register also specifies what is required to clear the specific interrupts. When the source of the interrupt is removed, reading the appropriate CLR_* register (for example, CLR_RX_OVER) clears the interrupt.

4.8.6.3 Built-in Two-Wire Serial Multiplexer

The device has built-in support for connecting to multiple two-wire serial devices that use the same two-wire serial address. This is done by using the multiplexed clock outputs (TWI_SCL_M*n*) for the two-wire serial devices rather than TWI_SCL. Depending on which device or devices it needs to talk to, software can then enable/disable the various clocks. This multiplexing feature is available for the first two-wire serial interface controller instance.

From the two-wire serial controllers point of view, it does not know if it is using TWI_SCL or TWI_SCL_M*n* clocks. When using multiplexed mode, software needs to enable/disable individual TWI_SCL_M*n* connections before initiating the access operation using the two wire serial controller.

Feedback on the clock (from slow two-wire serial devices) is automatically done for the TWI_SCL_M*n* lines that are enabled.

To enable multiplexed clocks, first configure the TWI_SCL_M*n* overlaid mode in the GPIO controller. Individual TWI_SCL_M*n* clocks are then enabled by writing 1 to the corresponding GPIO output bit (in DEVCPU_GCB::GPIO_OUT). To disable the clock, write 0 to the GPIO output bit. Disabled TWI_SCL_M*n* clocks are not driven during access and the feedback is disabled.

Note: In multiprocessor systems, the DEVCPU_GCB::GPIO_OUT_SET and DEVCPU_GCB::GPIO_OUT_CLR registers can be used to avoid race conditions.

4.8.7 MII Management Controller

This section provides information about the MII Management (MIIM) controllers. The following table lists the registers associated with the MII Management controllers.

Table 259 • **MIIM Registers**

Register	Description
DEVCPU_GCB::MII_STATUS	Controller status
DEVCPU_GCB::MII_CMD	Command and write data
DEVCPU_GCB::MII_DATA	Read data
DEVCPU_GCB::MII_CFG	Clock frequency configuration
DEVCPU_GCB::MII_SCAN_0	Auto-scan address range
DEVCPU_GCB::MII_SCAN_1	Auto-scan mask and expects
DEVCPU_GCB::MII_SCAN_LAST_RSLTS	Auto-scan result
DEVCPU_GCB::MII_SCAN_LAST_RSLTS_VLD	Auto-scan result valid
DEVCPU_GCB:: MII_SCAN_RSLTS_STICKY	Differences in expected versus read auto-scan

The device contains two MIIM controllers with equal functionality. Data is transferred on the MIIM interface using the Management Frame Format protocol specified in IEEE 802.3, Clause 22 or the MDIO Manageable Device protocol defined in IEEE 802.3, Clause 45. The Clause 45 protocol differs from the Clause 22 protocol by using indirect register access to increase the address range. The controller supports both Clause 22 and Clause 45. The first MIIM controller is connected to the internal PHYs of the device.

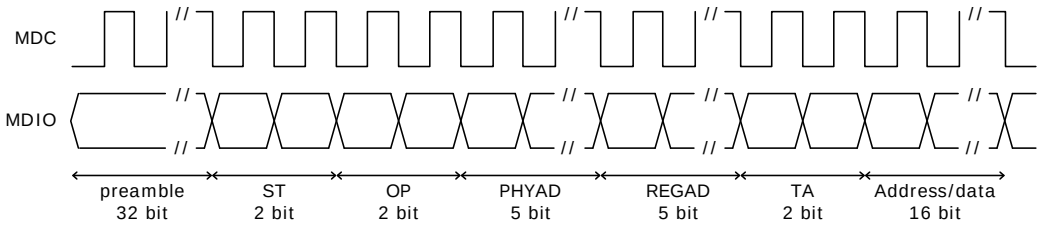
The MIIM interface pins for the second controller are overlaid functions on the GPIO interface. Before using this MIIM controller, the overlaid functions for the appropriate GPIO pins must first be enabled. For more information about overlaid functions on the GPIOs for these signals, see GPIO Overlaid Functions, page 439.

Table 260 • **MIIM Management Controller Pins**

Pin Name	I/O	Description
MDC1/GPIO	O	MIIM clock for controller 1
MDIO1/GPIO	I/O	MIIM data input/output for controller 1

The MDIO signal is changed or sampled on the falling edge of the MDC clock by the controller. The MDIO pin is tri-stated in-between access and when expecting read data.

Figure 161 • **MII Management Timing**



4.8.7.1 Clock Configuration

The frequency of the management interface clock generated by the MIIM controller is derived from the switch core's frequency. The MIIM clock frequency is configurable and is selected with MII_CFG.MIIM_CFG_PRESCALE. The calculation of the resulting frequency is explained in the register description for MII_CFG.MIIM_CFG_PRESCALE. The maximum frequency of the MIIM clock is 25 MHz.

4.8.7.2 MII Management PHY Access

Reads and writes across the MII management interface are performed through the MII_CMD register. Details of the operation, such as the PHY address, the register address of the PHY to be accessed, the operation to perform on the register (for example, read or write), and write data (for write operations), are set in the MII_CMD register. When the appropriate fields of MII_CMD are set, the operation is initiated by writing 0x1 to MII_CMD.MIIM_CMD_VLD. The register is automatically cleared when the MIIM command is initiated. When initiating single MIIM commands, MII_CMD.MIIM_CMD_SCAN must be set to 0x0.

When an operation is initiated, the current status of the operation can be read in MII_STATUS. The fields MII_STATUS.MIIM_STAT_PENDING_RD and MII_STATUS.MIIM_STAT_PENDING_WR can be used to poll for completion of the operation. For a read operation, the read data is available in MII_DATA.MIIM_DATA_RDDATA after completion of the operation. The value of MII_DATA.MIIM_DATA_RDDATA is only valid if MII_DATA.MIIM_DATA_SUCCESS indicates no read errors.

The MIIM controller contains a small command FIFO. Additional MIIM commands can be queued as long as MII_STATUS.MIIM_STAT_OPR_PEND is cleared. Care must be taken with read operations, because multiple queued read operations will overwrite MII_DATA.MIIM_DATA_RDDATA.

Note: A typical software implementation will never queue read operations, because the software needs read data before progressing the state of the software. In this case, MIIM_STATUS.MIIM_STAT_OPR_PEND is checked before issuing MIIM read or write commands, for read operations MII_STATUS.MIIM_STAT_BUSY is checked before returning read result.

By default, the MIIM controller operates in Clause 22 mode. To access Clause 45 compatible PHYs, MII_CFG.MIIM_ST_CFG_FIELD and MII_CMD.MIIM_CMD_OPR_FIELD must be set according to Clause 45 mode of operation.

4.8.7.3 PHY Scanning

The MIIM controller can be configured to continuously read certain PHY registers and detect if the read value is different from an expected value. If a difference is detected, a special sticky bit register is set or a CPU interrupt is generated, or both. For example, the controller can be programmed to read the status registers of one or more PHYs and detect if the Link Status changed since the sticky register was last read.

The reading of the PHYs is performed sequentially with the low and high PHY numbers specified in MII_SCAN_0 as range bounds. The accessed address within each of the PHYs is specified in MII_CMD.MIIM_CMD_REGAD. The scanning begins when a 0x1 is written to MII_CMD.MIIM_CMD_SCAN and a read operation is specified in MII_CMD.MIIM_CMD_OPR_FIELD. Setting MII_CMD.MIIM_CMD_SINGLE_SCAN stops the scanning after all PHYs have been scanned one time. The remaining fields of MII_CMD register are not used when scanning is enabled.

The expected value for the PHY register is set in MII_SCAN_1.MIIM_SCAN_EXPECT. The expected value is compared to the read value after applying the mask set in MII_SCAN_1.MIIM_SCAN_MASK. To don't-care a bit-position, write a 0 to the mask. If the expected value for a bit position differs from the read

value during scanning, and the mask register has a 1 for the corresponding bit, a mismatch for the PHY is registered.

The scan results from the most recent scan can be read in MII_SCAN_LAST_RSLTS. The register contains one bit for each of the possible 32 PHYs. A mismatch during scanning is indicated by a 0. MII_SCAN_LAST_RSLTS_VLD will indicate for each PHY if the read operation performed during the scan was successful. The sticky-bit register MII_SCAN_RSLTS_STICKY has the mismatch bit set for all PHYs that had a mismatch during scanning since the last read of the sticky-bit register. When the register is read, its value is reset to all-ones (no mismatches).

4.8.7.4 MII Management Interrupt

The MII management controllers can generate interrupts during PHY scanning. Each MII management controller has a separate interrupt signal to the interrupt controller. Interrupt is asserted when one or more PHYs have a mismatch during scan. The interrupt is cleared by reading the MII_SCAN_RSLTS_STICKY register, which resets all MII_SCAN_RSLTS_STICKY indications.

4.8.8 GPIO Controller

This section provides information about the use of GPIO pins.

The following table lists the registers associated with GPIO.

Table 261 • **GPIO Registers**

Register	Description
DEVCPU_GCB::GPIO_OUT	Value to drive on GPIO outputs
DEVCPU_GCB::GPIO_OUT_SET	Atomic set of bits in GPIO_OUT
DEVCPU_GCB::GPIO_OUT_CLR	Atomic clear of bits in GPIO_OUT
DEVCPU_GCB::GPIO_IN	Current value on the GPIO pins
DEVCPU_GCB::GPIO_OE	Enable of GPIO output mode (drive GPIOs)
DEVCPU_GCB::GPIO_ALT	Enable of overlaid GPIO functions
DEVCPU_GCB::GPIO_INTR	Interrupt on changed GPIO value
DEVCPU_GCB::GPIO_INTR_ENA	Enable interrupt on changed GPIO value
DEVCPU_GCB::GPIO_INTR_IDENT	Currently interrupting sources
DEVCPU_GCB::GPIO_SD_MAP	Mapping of parallel signal detects

The GPIO pins are individually programmable. GPIOs are inputs by default and can be individually changed to outputs through GPIO_OE. The value of the GPIO pins is reflected in the GPIO_IN register. GPIOs that are in output mode are driven to the value specified in GPIO_OUT.

In a system where multiple different CPU threads (or different CPUs) may work on the GPIOs at the same time, the GPIO_OUT_SET and GPIO_OUT_CLR registers provide a way for each thread to safely control the output value of individual GPIOs, without having to implement locked regions and semaphores.

The GPIO_ALT registers are only reset by external reset to the device. This means that software reset of the DEVCPU_GCB is possible without affecting the mapping of overlaid functions on the GPIOs.

4.8.8.1 GPIO Overlaid Functions

Most of the GPIO pins have overlaid (alternative) functions that can be enabled through replicated GPIO_ALT registers.

To enable a particular GPIO pin with the alternate function, set the G_ALT[*n*] register field in the replicated registers as follows:

- Overlaid mode 1, set GPIO_ALT[1][*n*], GPIO_ALT[0][*n*] = 1.
- Overlaid mode 2, set GPIO_ALT[1][*n*], GPIO_ALT[0][*n*] = 2.

- Normal GPIO mode, set GPIO_ALT[1][*n*], GPIO_ALT[0][*n*] = 0.

When the MIIM slave mode is enabled through the VCore_CFG strapping pins, specific GPIO pins are overtaken and used for the MIIM slave interface.

During reset, the VCore_CFG interface is sampled and used for VCore configuration. After reset, the device is released, and the GPIOs can be used for output or inputs. For more information, see VCore-III Configurations, page 393.

The following table maps the GPIO pins and available overlaid functions.

Table 262 • **GPIO Overlaid Functions**

Name	Overlaid Function 1	Overlaid Function 2	Overlaid Function 3	Configuration or Interface
GPIO_0	SG0_CLK			
GPIO_1	SG0_DO			REFCLK2_CONF0
GPIO_2	SG0_DI			
GPIO_3	SG0_LD			
GPIO_4	IRQ0_IN	IRQ0_OUT	TWI_SCL_M0	
GPIO_5	IRQ1_IN	IRQ1_OUT	TWI_SCL_M1	
GPIO_6	UART_RXD			
GPIO_7	UART_TXD			REFCLK2_CONF1
GPIO_8	SPI_nCS1	SFP0_SD	TWI_SCL_M2	
GPIO_9	PCI_Wake	SFP1_SD	SPI_nCS2	
GPIO_10	PTP_0	SFP2_SD	TWI_SCL_M3	
GPIO_11	PTP_1	SFP3_SD	TWI_SCL_M4	
GPIO_12	REF_CLK0	SFP4_SD	TWI_SCL_M5	
GPIO_13	REF_CLK1	SFP5_SD	TWI_SCL_M6	
GPIO_14	REF_CLK2	IRQ0_OUT	SPI_nCS3	
GPIO_15	REF_CLK3	IRQ1_OUT	SPI_nCS4	
GPIO_16	TACHO	SFP6_SD	SPI_nCS5	
GPIO_17	PWM		TWI_SCL_M7	
GPIO_18	PTP_2	SFP7_SD	SPI_nCS6	
GPIO_19	PTP_3	SFP8_SD	SPI_nCS7	
GPIO_20	UART2_RXD	SFP9_SD	SPI_nCS8	
GPIO_21	UART2_TXD			REFCLK2_CONF2
GPIO_22	MIIM1_MDC	SFP10_SD	TWI2_SDA	MIIM_SLV_MDC
GPIO_23	MIIM1_MDIO	SFP11_SD	TWI2_SCL	MIIM_SLV_MDIO
GPIO_24	TWI_SDA			
GPIO_25	TWI_SCL	SFP12_SD	TWI_SCL_M8	
GPIO_26	TWI_SCL_M9	SFP13_SD	SPI_nCS9	
GPIO_27	TWI_SCL_M10	SFP14_SD	SPI_nCS10	
GPIO_28	TWI_SCL_M11	SFP15_SD	SPI_nCS11	
GPIO_29	TWI_SCL_M12_AD			REFCLK2_SEL
GPIO_30	TWI_SCL_M13_AD			REFCLK0_CONF0

Table 262 • **GPIO Overlaid Functions *(continued)***

Name	Overlaid Function 1	Overlaid Function 2	Overlaid Function 3	Configuration or Interface
GPIO_31	TWI_SCL_M14_AD			REFCLK0_CONF1
GPIO_32	TWI_SCL_M15_AD			REFCLK0_CONF2
GPIO_33	RCVRD_CLK0			VCORE_CFG0
GPIO_34	RCVRD_CLK1			VCORE_CFG1
GPIO_35	RCVRD_CLK2			VCORE_CFG2
GPIO_36	RCVRD_CLK3			VCORE_CFG3

4.8.8.2 GPIO Interrupt

The GPIO controller continually monitors all inputs and set bits in the GPIO_INTR register whenever a GPIO changes its input value. By enabling specific GPIO pins in the GPIO_INTR_ENA register, a change indication from GPIO_INTR is allowed to propagate (as GPIO interrupt) from the GPIO controller to the VCore-III Interrupt Controller.

The currently interrupting sources can be read from GPIO_INTR_IDENT, this register is the result of a binary AND between the GPIO_INTR and GPIO_INTR_ENA registers.

4.8.8.3 Parallel Signal Detect

The GPIO controller has 16 programmable parallel signal detects, SFP0_SD through SFP15_SD. When parallel signal detect is enabled for a front port index, it overrides the signal-detect/loss-of-signal value provided by the serial GPIO controller.

To enable parallel signal detect *n*, first configure which port index from the serial GPIO controller that must be overwritten by setting GPIO_SD_MAP[*n*].G_SD_MAP, then enable the SFP*n*_SD function on the GPIOs.

The following table lists parallel signal detect pins, which are overlaid on GPIOs. For more information about overlaid functions on the GPIOs for these signals, see GPIO Overlaid Functions, page 439.

Table 263 • **Parallel Signal Detect Pins**

Register	I/O	Description
SFP0_SD/GPIO	I	Parallel signal detect 0
SFP1_SD/GPIO	I	Parallel signal detect 1
SFP2_SD/GPIO	I	Parallel signal detect 2
SFP3_SD/GPIO	I	Parallel signal detect 3
SFP4_SD/GPIO	I	Parallel signal detect 4
SFP5_SD/GPIO	I	Parallel signal detect 5
SFP6_SD/GPIO	I	Parallel signal detect 6
SFP7_SD/GPIO	I	Parallel signal detect 7
SFP8_SD/GPIO	I	Parallel signal detect 8
SFP9_SD/GPIO	I	Parallel signal detect 9
SFP10_SD/GPIO	I	Parallel signal detect 10
SFP11_SD/GPIO	I	Parallel signal detect 11
SFP12_SD/GPIO	I	Parallel signal detect 12
SFP13_SD/GPIO	I	Parallel signal detect 13
SFP14_SD/GPIO	I	Parallel signal detect 14

Table 263 • **Parallel Signal Detect Pins *(continued)***

Register	I/O	Description
SFP15_SD/GPIO	I	Parallel signal detect 15

4.8.9 Serial GPIO Controller

The device features one serial GPIO (SIO) controller. By using a serial interface, the SIO controller significantly extends the number of available GPIOs with a minimum number of additional pins on the device. The primary purpose of the SIO controller is to connect control signals from SFP modules and to act as an LED controller.

Each SIO controller supports up to 128 serial GPIOs (SGPIOs) organized into 32 ports, with four SGPIOs per port.

The following table lists the registers associated with the SIO controller.

Table 264 • **SIO Registers**

Register	Description
DEVCPU_GCB::SIO_INPUT_DATA	Input data
DEVCPU_GCB::SIO_CFG	General configuration
DEVCPU_GCB::SIO_CLOCK	Clock configuration
DEVCPU_GCB::SIO_PORT_CFG	Output port configuration
DEVCPU_GCB::SIO_PORT_ENA	Port enable
DEVCPU_GCB::SIO_PWM_CFG	PWM configuration
DEVCPU_GCB::SIO_INTR_POL	Interrupt polarity
DEVCPU_GCB::SIO_INTR_RAW	Raw interrupt status
DEVCPU_GCB::SIO_INTR_TRIGGER0	Interrupt trigger mode 0 configuration
DEVCPU_GCB::SIO_INTR_TRIGGER1	Interrupt trigger mode 1 configuration
DEVCPU_GCB::SIO_INTR	Currently interrupting SGPIOs
DEVCPU_GCB::SIO_INTR_ENA	Interrupt enable
DEVCPU_GCB::SIO_INTR_IDENT	Currently active interrupts

The following table lists the SIO controller pins, which are overlaid functions on GPIO pins. For more information about overlaid functions on the GPIOs for these signals, see GPIO Overlaid Functions, page 439.

Table 265 • **SIO Controller Pins**

Pin Name	I/O	Description
SG0_CLK/GPIO	O	SIO clock output, frequency is configurable using SIO_CLOCK.SIO_CLK_FREQ.
SG0_DO/GPIO	O	SIO data output.
SG0_DI/GPIO	I	SIO data input.
SG0_LD/GPIO	O	SIO load data, polarity is configurable using SIO_CFG.SIO_LD_POLARITY.

The SIO controller works by shifting SGPIO values out on SG0_DO though a chain of shift registers on the PCB. After shifting a configurable number of SGPIO bits, the SIO controller asserts SG0_LD, which causes the shift registers to apply the values of the shifted bits to outputs. The SIO controller can also read inputs while shifting out SGPIO values on SG0_DO by sampling the SG0_DI input. The values sampled on SG0_DI are made available to software.

If the SIO controller is only used for outputs, the use of the load signal is optional. If the load signal is omitted, simpler shift registers (without load) can be used, however, the outputs of these registers will toggle during shifting.

When driving LED outputs, it is acceptable that the outputs will toggle when SGPIO values are updated (shifted through the chain). When the shift frequency is fast, the human eye is not able to see the shifting though the LEDs.

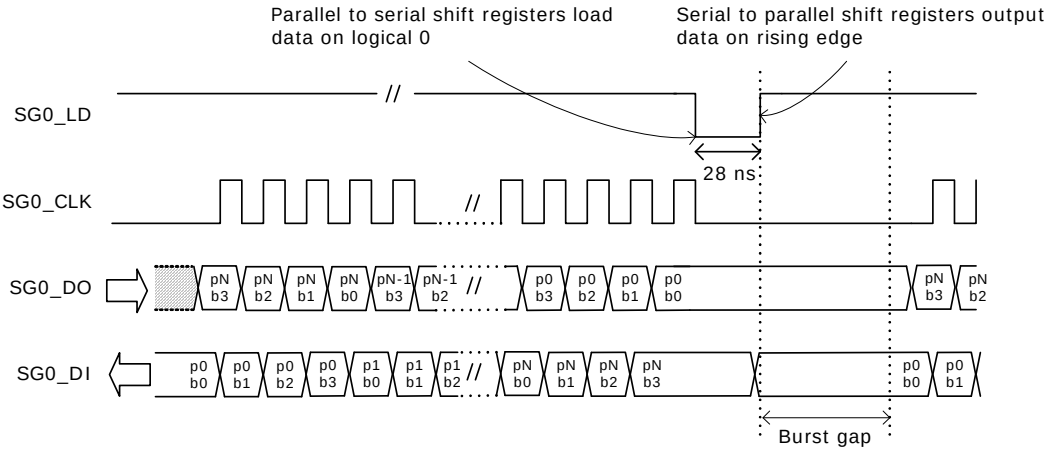
The number of shift registers in the chain is configurable. The SIO controller allows enabling of individual ports through SIO_PORT_ENA; only enabled ports are shifted out on SG0_DO. Ports that are not enabled are skipped during shifting of GPIO values.

Note: SIO_PORT_ENA allows skipping of ports in the SGPIO output stream that are not in use. The number of GPIOs per (enabled) port is configurable as well, through SIO_CFG.SIO_PORT_WIDTH this can be set to 1, 2, 3, or 4 bits. The number of bits per port is common for all enabled ports, so the number of shift registers on the PCB must be equal to the number of enabled ports times the number of SGPIOs per port.

Enabling of ports and configuration of SGPIOs per port applies to both output mode and input mode. Unlike a regular GPIO port, a single SGPIO position can be used both as output and input. That is, software can control the output of the shift register AND read the input value at the same time. Using SGPIOs as inputs requires load-capable shift registers.

Regular shift registers and load-capable shift-registers can be mixed, which is useful when driving LED indications for integrated PHYs while supporting reading of link status from SFP modules, for example.

***Figure 162* • SIO Timing**



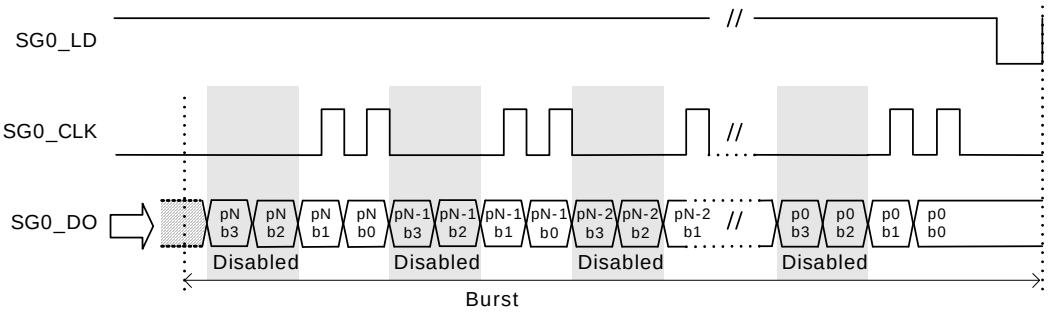
The SGPIO values are output in bursts followed by assertion of the SG0_LD signal. Values can be output as a single burst or as continuous bursts separated by a configurable burst gap. The maximum length of a burst is 32 × 4 data cycles. The burst gap is configurable in steps of approximately 1 ms, from 0 ms to 33 ms through SIO_CFG.SIO_BURST_GAP_DIS and SIO_CFG.SIO_BURST_GAP.

A single burst is issued by setting SIO_CFG.SIO_SINGLE_SHOT. The field is automatically cleared by hardware when the burst is finished. To issue continuous bursts, set SIO_CFG.SIO_AUTO_REPEAT. The SIO controller continues to issue bursts until SIO_CFG.SIO_AUTO_REPEAT is cleared.

SGPIO output values are configured in SIO_PORT_CFG.BIT_SOURCE. The input value is available in SIO_INPUT_DATA.

The following illustration shows what happens when the number of SGPIOs per port is configured to two (through SIO_CFG.SIO_PORT_WIDTH). Disabling ports (through SIO_PORT_ENA) is handled in the same way as disabling the SGPIO ports.

Figure 163 • **SIO Timing with SGPIOs Disabled**



The frequency of the SG0_CLK clock output is configured through SIO_CLOCK.SIO_CLK_FREQ. The SG0_LD output is asserted after each burst; this output is asserted for a period of 25 ns to 30 ns. The polarity of SG0_LD is configured in SIO_CFG.SIO_LD_POLARITY.

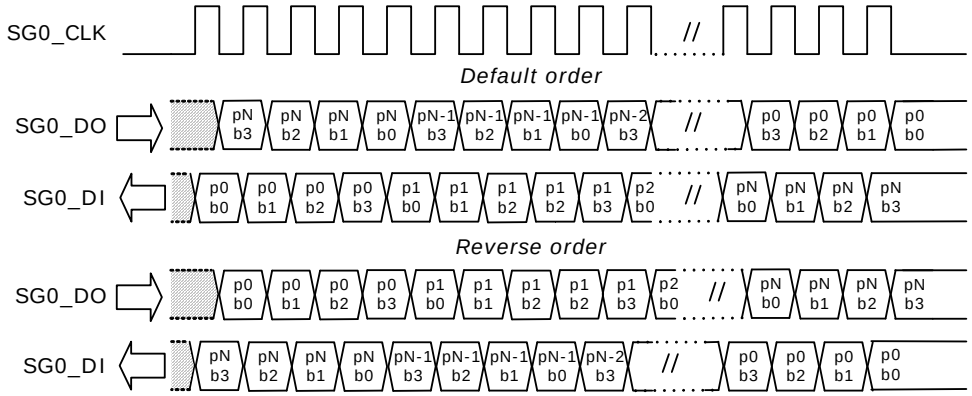
The SG0_LD output can be used to ensure that outputs are stable when serial data is being shifted through the registers. This can be done by using the SG0_LD output to shift the output values into serial-to-parallel registers after the burst is completed. If serial-to-parallel registers are not used, the outputs will toggle while the burst is being shifted through the chain of shift registers. A universal serial-to-parallel shift register outputs the data on a positive-edge load signal, and a universal parallel-to-serial shift register shifts data when the load pin is high, so one common load signal can be used for both input and output serial parallel conversion.

The assertion of SG0_LD happens after the burst to ensure that after power up, the single burst will result in well-defined output registers. Consequently, to sample input values one time, two consecutive bursts must be issued. The first burst results in the input values being sampled by the serial-to-parallel registers, and the second burst shifts the input values into the SIO controller.

The port order required in the serial bitstream depends on the physical layout of the shift register chain. Often the input and output port orders must be opposite in the serial streams. The port order of the input and output bitstream is independently configurable in SIO_CFG.SIO_REVERSE_INPUT and SIO_CFG.SIO_REVERSE_OUTPUT.

The following illustration shows the port order.

Figure 164 • **SGPIO Output Order**



4.8.9.1 Output Modes

The output mode of each SGPIO can be individually configured in SIO_PORT_CFG.BIT_SOURCE. The SIO controller features three output modes:

- Static
- Blink
- Link activity

The output mode can additionally be modified with PWM (SIO_PORT_CFG.PWM_SOURCE) and configurable polarity (SIO_PORT_CFG.BIT_POLARITY).

Static Mode The static mode is used to assign a fixed value to the SGPIO, for example, fixed 0 or fixed 1.

Blink Mode The blink mode makes the SGPIO blink at a fixed rate. The SIO controller features two blink modes that can be set independently. A SGPIO can then be configured to use either blink mode 0 or blink mode 1. The blink outputs are configured in SIO_CFG.SIO_BMODE_0 and SIO_CFG.SIO_BMODE_1. To synchronize the blink modes between different devices, reset the blink counter using SIO_CFG.SIO_BLINK_RESET. All the SIO controllers on a device must be reset at same time to maintain the synchronization. The burst toggle mode of blink mode 1 toggles the output with every burst.

Table 266 • **Blink Modes**

Register	Description
Blink Mode 0	0: 20 Hz blink frequency 1: 10 Hz blink frequency 2: 5 Hz blink frequency 3: 2.5 Hz blink frequency
Blink Mode 1	0: 20 Hz blink frequency 1: 10 Hz blink frequency 2: 5 Hz blink frequency 3: Burst toggle

Link Activity Mode The link activity mode makes the output blink when there is activity on the port module (Rx or Tx). The following table lists the mapping between the SIO port number and port modules.

Table 267 • **SIO Controller Port Mapping**

SIO Port	SG0 Mapping
0	Port module 0
1	Port module 1
2	Port module 2
3	Port module 3
4	Port module 4
5	Port module 5
6	Port module 6
7	Port module 7
8	Port module 8
9	Port module 9
10	Port module 10
11	Free
12	Free
13	Free
14	Free
15	Free
16	Free
17	Free

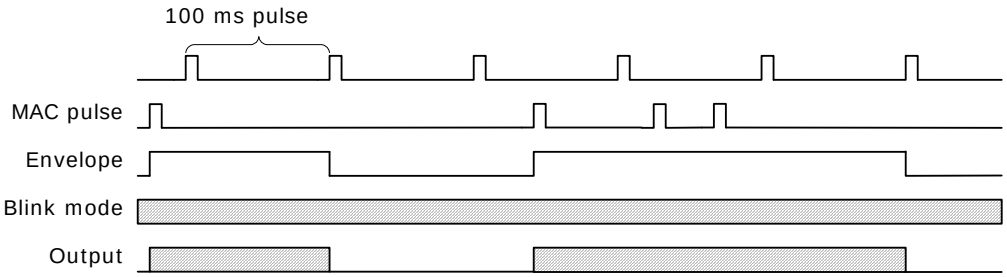
Table 267 • **SIO Controller Port Mapping *(continued)***

SIO Port	SG0 Mapping
18	Free
19	Free
20	Free
21	Free
22	Free
23	Free
24	Free
25	Free
26	Free
27	Free
28	Free
29	Free
30	Free
31	Free

The link activity mode uses an envelope signal to gate the selected blinking pattern (blink mode 0 or blink mode 1). When the envelope signal is asserted, the output blinks, and when the envelope pattern is de-asserted, the output is turned off. To ensure that even a single packet makes a visual blink, an activity pulse from the port module is extended to minimum 100 ms. If another packet is sent while the envelope signal is asserted, the activity pulse is extended by another 100 ms. The polarity of the link activity modes can be set in SIO_PORT_CFG.BIT_SOURCE.

The following illustration shows the link activity timing.

Figure 165 • **Link Activity Timing**



4.8.9.2 SIO Interrupt

The SIO controller can generate interrupts based on the value of the input value of the SGPIOs. All interrupts are level sensitive.

Interrupts are enabled using the two registers. Interrupts can be individually enabled for each port in SIO_INTR_ENA (32 bits) and in SIO_CFG.SIO_GPIO_INTR_ENA (4 bits) interrupts are enabled for the four inputs per port. In other words, SIO_CFG.SIO_GPIO_INTR_ENA is common for all 32 ports.

The SIO controller has four interrupt registers that each has one bit for each of the 128 GPIOs:

- SIO_INTR_RAW is high if the corresponding input bit is high (corrected for polarity as configured in SIO_INTR_POL). This register changes when the input changes.
- SIO_INTR is high if the condition of the configured trigger mode for the bit is met. The trigger mode can be configured in SIO_INTR_TRIGGER0 and SIO_INTR_TRIGGER1 between level-triggered,

edge-triggered, falling-edge-triggered, and rising-edge-triggered interrupt. This register is a sticky bit vector and can only be cleared by software. A bit is cleared by writing a 1 to the bit position.

- SIO_INTR_IDENT is the result of SIO_INTR with the disabled interrupts (from SIO_INTR_ENA and SIO_GPIO_INTR_ENA) removed. This register changes when SIO_INTR or the enable registers change.

The SIO controller has one interrupt output connected to the main interrupt controller, which is asserted when one or more interrupts in SIO_INTR_IDENT are active. To determine which SGPIO is causing the interrupt, the CPU must read this register. The interrupt output remains high until all interrupts in SIO_INTR_IDENT are cleared (either by clearing SIO_INTR or disabling the interrupts in SIO_INTR_ENA and SIO_GPIO_INTR_ENA).

4.8.9.3 Loss of Signal Detection

The SIO controller can propagate loss of signal detection inputs directly to the signal detection input of the port modules. This is useful when, for example, SFP modules are connected to the device. The mapping between SIO ports and port modules is the same as for the link activity outputs; port 0 is connected to port module 0, port 1 is connected to port module 1, and so on.

The value of SGPIO bit 0 of each SIO port is forwarded directly to the loss of signal input on the corresponding port module. The port module must enable the loss of signal input locally.

Loss of signal can also be taken directly from overlaid functions on the regular GPIOs. In that case, the input from the SIO controller is ignored.

The polarity of the loss of signal input is configured using SIO_INT_POL, meaning the same polarity must be used for loss of signal detect and interrupt.

4.8.10 Fan Controller

The device includes a fan controller that can be used to control and monitor a system fan. A pulse width modulation (PWM) output regulates the fan speed. The fan speed is monitored using a TACHO input. The fan controller is especially powerful when combined with the internal temperature sensor. For more information, see Temperature Sensor, page 448.

The following table lists the registers associated with the fan controller.

Table 268 • **Fan Controller**

Register	Description
DEVCPU_GCB::FAN_CFG	General configuration
DEVCPU_GCB::FAN_CNT	Fan revolutions counter

The following table lists fan controller pins, which are overlaid on GPIOs. For more information about overlaid functions on the GPIOs for these signals, see GPIO Overlaid Functions, page 439.

Table 269 • **Fan Controller Pins**

Register	I/O	Description
TACHO/GPIO	I	TACHO input for counting revolutions
PWM/GPIO	O	PWM fan output

The PWM output can be configured to any of the following frequencies in FAN_CFG.PWM_FREQ: 10 Hz, 20 Hz, 40 Hz, 60 Hz, 80 Hz, 100 Hz, 120 Hz, or 25 kHz.

The low frequencies can be used for driving three-wire fans using a FET/transistor. The 25 kHz frequency can be used for four-wire fans that use the PWM input internally to control the fan. The duty cycle of the PWM output is programmable from 0% to 100%, with 8-bit accuracy. The polarity of the output can be controlled by FAN_CFG.INV_POL, so a duty-cycle of 100%, for example, can be either always low or always high.

The PWM output pin can be configured to act as a normal output or as an open-collector output, where the output value of the pin is kept low, but the output enable is toggled. The open-collector output mode is enabled by setting FAN_CFG.PWM_OPEN_COL_ENA.

Note: By using open-collector mode, it is possible to externally pull-up to a higher voltage than the maximum GPIO I/O supply. The GPIO pins are 3.3V-tolerable.

The speed of the fan is measured using a 16-bit counter that counts the rising edges on the TACHO input. A fan usually gives one to four pulses per revolution, depending on the fan type. The counter value is available in the FAN_CNT register. Depending on the value of FAN_CFG.FAN_STAT_CFG, the FAN_CNT register is updated in two different ways:

- If FAN_CFG.FAN_STAT_CFG is set, the FAN_CNT register behaves as a 16-bit wrapping counter that shows the total number of ticks on the TACHO input.
- If FAN_CFG.FAN_STAT_CFG is cleared, the FAN_CNT register is updated one time per second with the number of TACHO ticks received during the last second.

Optionally, the TACHO input is gated by the polarity-corrected PWM output by setting FAN_CFG.GATE_ENA, so that only TACHO pulses received while the polarity corrected PWM output is high are counted. Glitches on the TACHO input can occur right after the PWM output goes high. As a result, the gate signal is delayed by 10 µs when PWM goes high. There is no delay when PWM goes low, and the length of the delay is not configurable. Software must read the counter value in FAN_CNT and calculate the RPM of the fan.

An example of how to calculate the RPM of the fan is if the fan controller is configured to 100 Hz and a 20% duty cycle, each PWM pulse is high in 2 ms and low in 8 ms. If gating is enabled, the gating of the TACHO input is open in 1.99 ms and closed in 8.01 ms. If the fan is turning with 100 RPMs and gives two TACHO pulses per revolution, it will ideally give 200 pulses per minute. TACHO pulses are only counted in 19.99% of the time, so it will give 200 × 0.1999 = 39.98 pulses per minute. If the additional 10 µs gating time is ignored, the counter value is multiplied by 2.5 to get the RPM value, because there is a 20% duty cycle with two TACHO pulses per revolution. By multiplying with 2.5, the RPM value is calculated to 99.95, which is 0.05% off the correct value (due to the 10 µs gating time).

4.8.11 Temperature Sensor

This section provides information about the on-die temperature sensor. When enabled the temperature sensor logic will continually monitor the temperature of the die and make this available for software.

The following table lists the registers associated with the temperature monitor.

Table 270 • **Temperature Sensor Registers**

Register	Description
HSIO::TEMP_SENSOR_CTRL	Enabling of sensor
HSIO::TEMP_SENSOR_STAT	Temperature value

The temperature sensor is enabled by setting TEMP_SENSOR_CTRL.SAMPLE_ENA. The temperature sensor then samples the temperature every 500 µs and shows the current temperature in TEMP_SENSOR_STAT. The formula for converting TEMP field value to centigrade temperature is:

Temp (°C) = 177.4 – 0.8777 × TEMP_SENSOR_STATE.TEMP

It takes approximately 500 µs after setting SAMPLE_ENA until the first temperature sample is ready. The TEMP_SENSOR_STAT.TEMP_VALID field is set when the temperature value is available.

4.8.12 Memory Integrity Monitor

Soft errors happen in all integrated circuits as a result of natural alpha decay, cosmic radiation, or electrical disturbances in the environment in which the device operates. The chance of soft errors happening in a memory (RAM) is higher than for flip-flop based logic, because the memory structures are physically small and changes require less outside force than in flip flops. The device has built-in protection from soft errors by using error correcting code (ECC) on critical memories. In addition, the device allows monitoring and reporting of soft error events.

The following table lists the registers associated with the memory integrity monitor.

Table 271 • **Integrity Monitor Registers**

Register	Description
DEVCPU_GCB::MEMITGR_CTRL	Trigger monitor state changes.
DEVCPU_GCB::MEMITGR_STAT	Current state of the monitor and memory status.
DEVCPU_GCB::MEMITGR_INFO	Shows indication when in DETECT state.
DEVCPU_GCB::MEMITGR_IDX	Shows memory index when in DETECT state.
DEVCPU_GCB::MEMITGR_DIV	Monitor speed.

The memory integrity monitor looks for memory soft-error indications. Correctable (single bit) and non-correctable (multibit or parity) indications are detected during memory read and can be reported to software by the ITGR software interrupt. For information about how to enable the this interrupt, see Interrupt Controller, page 451.

The memory integrity monitor operates in three different states: IDLE, LISTEN, and DETECT. After a reset, the monitor starts in the IDLE state.

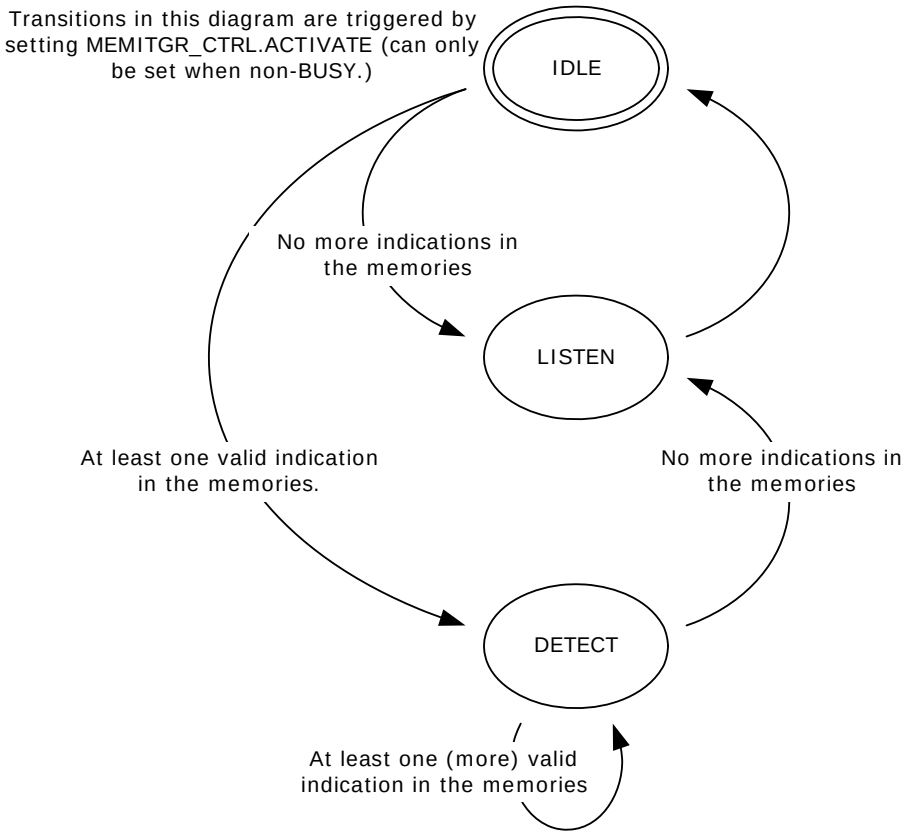
IDLE The monitor is deactivated and in quiet mode. In IDLE mode, the memories still correct, detect, and store indications locally, but they are not able to report indications to the monitor.

LISTEN In the LISTEN state, the monitor looks for indications in the memories. MEMITGR_STAT.INDICATION is set (and interrupt is asserted) when indications are detected.

DETECT DETECT state is used when indications are read from the memories. It means that a valid indication is available in MEMITGR_INFO and the corresponding memory index in MEMITGR_IDX.

The current state of the monitor is reported in MEMITGR_STAT.MODE_IDLE, MEMITGR_STAT.MODE_DETECT, and MEMITGR_STAT.MODE_LISTEN. Software initiates transitions between states by setting the one-shot MEMITGR_CTRL.ACTIVATE field. It may take some time to transition from one state to the next. The MEMITGR_CTR.ACTIVATE field is not cleared before the next state is reached (also the MEMITGR_STAT.MODE_BUSY field is set while transitioning between states).

***Figure 166* • Monitor State Diagram**



The first time after reset that MEMITGR_CTRL.ACTIVATE is set, the monitor resets the detection logic in all the memories and transitions directly from IDLE to LISTEN state.

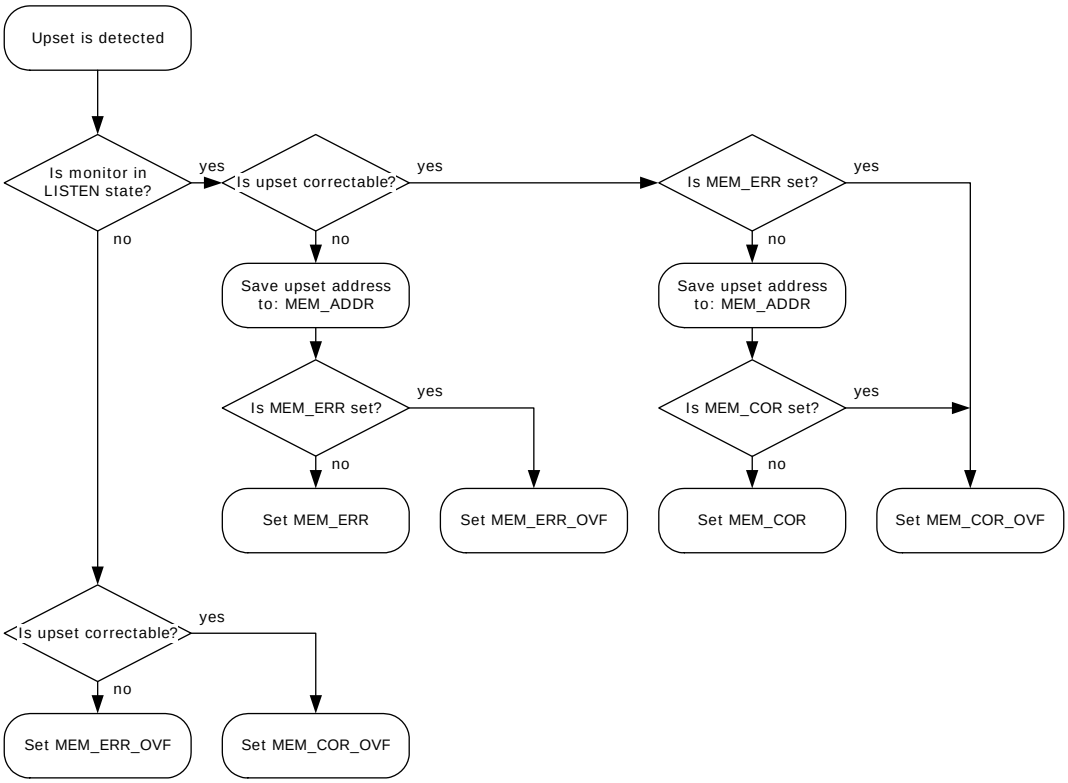
Before setting MEMITGR_CTRI.ACTIVATE for the first time, speed up the monitor by setting MEMITGR_DIV.MEM_DIV to the value specified in the registers list. The memories of the PCIe MAC are clock-gated and bypassed when PCIe interface is not enabled; enable monitoring of PCIe MAC memories by setting ICPU_CFG::PCIE_CFG.MEM_RING_CORE_ENA. This field must not be set when PCIe interface is not enabled.

To read out indications, first transition from LISTEN to IDLE, then continue transitioning until the LISTEN state is reached. Every time the monitor ends up in the DETECT state, an indication is available in the MEMITGR_INFO and MEMITGR_IDX registers. Each memory stores one indication. Indications are cleared when they are read by way of the monitor. Each indication contains four flags and one memory address.

- The MEM_ERR flag is set when a non-correctable upset is detected and the corresponding address is available in MEM_ADDR.
- The MEM_ERR_OVF flag is set when a non-correctable upset is detected for which address could not be stored.
- The MEM_COR flag is set when a correctable upset is detected and the corresponding address is available in MEM_ADDR.
- The MEM_COR_OVF flag is set when a correctable upset is detected for which address could not be stored.

Information about non-correctable upsets is prioritized over correctable upsets. Address can only be saved when the monitor is in LISTEN mode. The following flowchart shows how the detection logic sets flags and address.

Figure 167 **• Memory Detection Logic**



If the MEM_ERR_OVF or MEM_COR_OVF flag is set, at least one event has occurred for which the address could not be stored.

The following table shows ECC-enabled memories in the device, their index, and the recommended approach for handling indications. If the controller reports an index that is not mentioned in the list, the recommended approach is to reboot the device.

Table 272 **• Memories with Integrity Support**

Index	Description
Others	For unlisted indexes, the recommended approach is to reboot the device.

The VCore-III CPU implements parity protection of all VCore-III cache structures as specified by MIPS architecture. For information about how to enable parity protection and interrupt on parity upsets, see the MIPS documentation.

Reading from uninitialized memory locations has a high plausibility of triggering non correctable or correctable indications. This is useful when developing integrity monitor software driver. For example, powering up a system without initializing the VCAPs and reading actions and sticky bits will trigger monitor indications. Note that the contents of memories are not changed by device reset, so power cycle is needed to reset the memories.

4.8.13 Interrupt Controller

This section provides information about the VCore-III interrupt controller.

The following table lists the registers associated with the interrupt controller.

Table 273 • **Interrupt Controller Registers**

Register	Description
ICPU_CFG::INTR_RAW	Current value of interrupt inputs
ICPU_CFG::INTR_BYPASS	Forces non-sticky function
ICPU_CFG::INTR_TRIGGER	Configures edge or level sensitive events
ICPU_CFG::INTR_FORCE	Forces events (for software debug)
ICPU_CFG::INTR_STICKY	Currently logged events
ICPU_CFG::INTR_ENA	Enables interrupt sources
ICPU_CFG::INTR_ENA_CLR	Atomic clear of bits in INTR_ENA
ICPU_CFG::INTR_ENA_SET	Atomic set of bits in INTR_ENA
ICPU_CFG::INTR_IDENT	Currently enabled and interrupting sources
ICPU_CFG::DST_INTR_MAP	Mapping of interrupt sources to destinations
ICPU_CFG::DST_INTR_IDENT	Currently enabled, mapped, and interrupting sources per destination
ICPU_CFG::DEV_INTR_POL	Polarity of module interrupt inputs
ICPU_CFG::DEV_INTR_RAW	Current value of module interrupts
ICPU_CFG::DEV_INTR_BYPASS	Forces non-sticky function for module interrupts
ICPU_CFG::DEV_INTR_TRIGGER	Configures edge or level sensitive events for module interrupts
ICPU_CFG::DEV_INTR_STICKY	Currently logged module interrupt events
ICPU_CFG::DEV_INTR_ENA	Enables module interrupts
ICPU_CFG::DEV_INTR_IDENT	Currently interrupting and enabled module interrupts
ICPU_CFG::EXT_SRC_INTR_POL	Polarity of external interrupt inputs.
ICPU_CFG::EXT_DST_INTR_POL	Polarity of external interrupt outputs.
ICPU_CFG::EXT_DST_INTR_DRV	Drive mode for external interrupt outputs

The interrupt controller maps interrupt sources from VCore-III and switch core blocks, port modules, and external interrupt inputs to four interrupt destinations. Two interrupt destinations are mapped to the VCore-III CPU, and two can be transmitted from the device using the overlaid functions on GPIOs or using PCIe inband interrupt signaling.

The following table lists the available interrupt sources in the device.

Table 274 • **Interrupt Sources**

Source Name	Description
DEV	Aggregated port module interrupt. This interrupt is asserted if there is an active and enabled interrupt from any of the device's port modules. See Port Module Interrupts. This interrupt has bit index 0 in INTR_* and DST_INTR_* registers.
EXT_SRC0	External interrupt source 0. See External Interrupts. This interrupt has bit index 1 in INTR_* and DST_INTR_* registers.
EXT_SRC1	External interrupt source 1. See External Interrupts. This interrupt has bit index 2 in INTR_* and DST_INTR_* registers.
TIMER0	Timer 0 interrupt. See Timers. This interrupt has bit index 3 in INTR_* and DST_INTR_* registers.

Table 274 • **Interrupt Sources *(continued)***

Source Name	Description
TIMER1	Timer 1 interrupt. See Timers. This interrupt has bit index 4 in INTR_* and DST_INTR_* registers.
TIMER2	Timer 2 interrupt. See Timers. This interrupt has bit index 5 in INTR_* and DST_INTR_* registers.
UART	UART interrupt. See UART Interrupt. This interrupt has bit index 6 in INTR_* and DST_INTR_* registers.
UART2	UART2 interrupt. See UART Interrupt. This interrupt has bit index 7 in INTR_* and DST_INTR_* registers.
TWI	TWI interrupt. See Two-Wire Serial Interface Interrupt. This interrupt has bit index 8 in INTR_* and DST_INTR_* registers.
TWI2	TWI2 interrupt. See Two-Wire Serial Interface Interrupt. This interrupt has bit index 9 in INTR_* and DST_INTR_* registers.
SIMC	Serial Master Controller interrupt. See Serial Master Controller Interrupt. This interrupt has bit index 10 in INTR_* and DST_INTR_* registers.
SW0	Software interrupt 0. See Mailbox and Semaphores. This interrupt has bit index 11 in INTR_* and DST_INTR_* registers.
SW1	Software interrupt 1. See Mailbox and Semaphores. This interrupt has bit index 12 in INTR_* and DST_INTR_* registers.
SGPIO0	Serial GPIO interrupt 0. See SIO Interrupt. This interrupt has bit index 13 in INTR_* and DST_INTR_* registers.
GPIO	Parallel GPIO interrupt. See GPIO Interrupt. This interrupt has bit index 14 in INTR_* and DST_INTR_* registers.
MIIM0	MIIM Controller 0 interrupt. See MII Management Interrupt. This interrupt has bit index 15 in INTR_* and DST_INTR_* registers.
MIIM1	MIIM Controller 1 interrupt. See MII Management Interrupt. This interrupt has bit index 16 in INTR_* and DST_INTR_* registers.
FDMA	Frame DMA interrupt, see FDMA Events and Interrupts. This interrupt has bit index 17 in INTR_* and DST_INTR_* registers.
ANA	Analyzer interrupt. See Interrupt Handling. This interrupt has bit index 18 in INTR_* and DST_INTR_* registers.
PTP_RDY	Time stamp ready interrupt. See Hardware Time Stamping Module. This interrupt has bit index 19 in INTR_* and DST_INTR_* registers.
PTP_SYNC	PTP synchronization interrupt. See Master Timer. This interrupt has bit index 20 in INTR_* and DST_INTR_* registers.
ITGR	Memory integrity interrupt. See Memory Integrity Monitor, page 448. This interrupt has bit index 21 in INTR_* and DST_INTR_* registers.
XTR_RDY	Extraction data ready interrupt. See Frame Extraction. This interrupt has bit index 22 in INTR_* and DST_INTR_* registers.
INJ_RDY	Injection ready interrupt. See Frame Injection. This interrupt has bit index 23 in INTR_* and DST_INTR_* registers.
PCIE	PCIe interrupt. See Power Management. This interrupt has bit index 24 in INTR_* and DST_INTR_* registers.
OAM_VOP	OAM/VOP interrupt. See Interrupt Controller. This interrupt has bit index 25 in INTR_* and DST_INTR_* registers.

The following table lists the available interrupt destinations in the device.

Table 275 • **Interrupt Destinations**

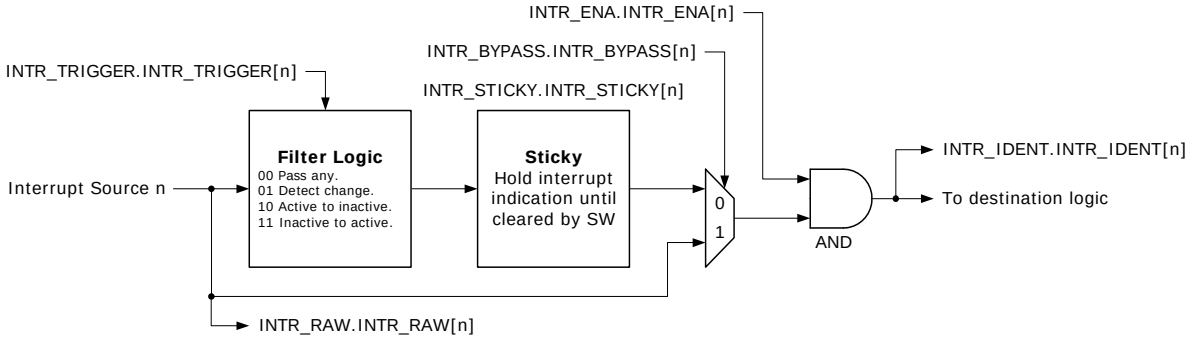
Destination Name	Description
CPU0	Interrupt 0 to VCore-III CPU. This interrupt has replication index 0 in DST_INTR_* registers.
CPU1	Interrupt 1 to VCore-III CPU. This interrupt has replication index 1 in DST_INTR_* registers.
EXT_DST0	External interrupt destination 0. See External Interrupts. This interrupt has replication index 2 in DST_INTR_* registers.
EXT_DST1	External interrupt destination 1. See External Interrupts. This interrupt has replication index 3 in DST_INTR_* registers.

All interrupts, events, and indications inside in the interrupt controller are active high. If an interrupt source supports polarity correction, it is applied before going into the interrupt controller. If an interrupt destination supports polarity correction, it is applied after leaving the interrupt controller.

4.8.13.1 Interrupt Source Configuration

Interrupt sources are handled identically inside the interrupt controller. This section describes interrupt source n, which refers to the bit index of that interrupt source in the INTR_* and DST_INTR_* registers. The following illustration shows the logic associated with a single interrupt source.

Figure 168 • **Interrupt Source Logic**



The current value of an interrupt source is available in INTR_RAW.INTR_RAW[*n*]. INTR_STICKY.INTR_STICKY[*n*] is set when the interrupt controller detects an interrupt. There are two detection methods:

- When INTR_TRIGGER.INTR_TRIGGER[*n*] is set to level-activated, the interrupt controller continually sets INTR_STICKY.INTR_STICKY[*n*] for as long as the interrupt source is active.
- When INTR_TRIGGER.INTR_TRIGGER[*n*] is set to edge-triggered, the interrupt controller only sets INTR_STICKY.INTR_STICKY[*n*] when the interrupt source changes value.
- When INTR_TRIGGER.INTR_TRIGGER[*n*] is set to falling-edge-triggered, the interrupt controller only sets INTR_STICKY.INTR_STICKY[*n*] when the interrupt source changes from active to inactive value.
- When INTR_TRIGGER.INTR_TRIGGER[*n*] is set to rising-edge-triggered, the interrupt controller only sets INTR_STICKY.INTR_STICKY[*n*] when the interrupt source changes from inactive to active value.

Software can clear INTR_STICKY.INTR_STICKY[*n*] by writing 1 to bit n. However, the interrupt controller will immediately set this bit again if the source input is still active (when INTR_TRIGGER is 0) or if it sees a triggering event on the source input (when INTR_TRIGGER different from 0).

The interrupt source is enabled in INTR_ENA.INTR_ENA[*n*]. When INTR_STICKY.INTR_STICKY[*n*] is set and the interrupt is enabled, the interrupt is indicated towards the interrupt destinations. For more

information, see Interrupt Destination Configuration. An active and enabled interrupt source sets INTR_IDENT.INTR_IDENT[*n*].

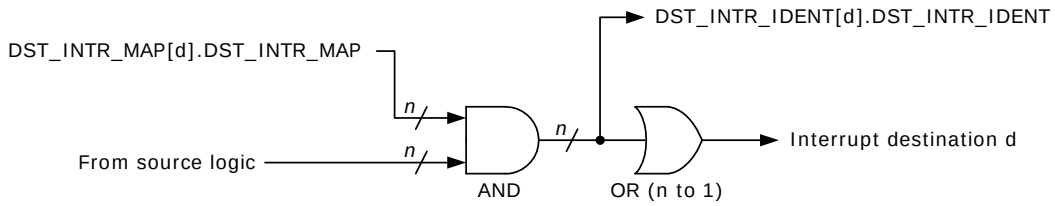
On rare occasions it is desirable to bypass the stickiness of interrupt sources and use INTR_RAW.INTR_RAW[*n*] directly instead of INTR_STICKY.INTR_STICKY[*n*]. Set INTR_BYPASS.INTR_BYPASS[*n*] to enable bypass and ignore INTR_STICKY and INTR_TRIGGER configurations.

Note: The bypass function may be useful for some software interrupt handler architectures. It should only be used for interrupt sources that are guaranteed to be sticky in the source block. For example, the GPIO interrupts that are generated from sticky bits in DEVCPU_GCB::GPIO_INTR may be applicable for the bypass mode.

4.8.13.2 Interrupt Destination Configuration

The four interrupt destinations are handled identically in the interrupt controller. This section describes destination d, which refers to the replication index of that interrupt in the DST_INTR_* registers. The following illustration shows the logic associated with a single interrupt destination.

***Figure 169* • Interrupt Destination Logic**



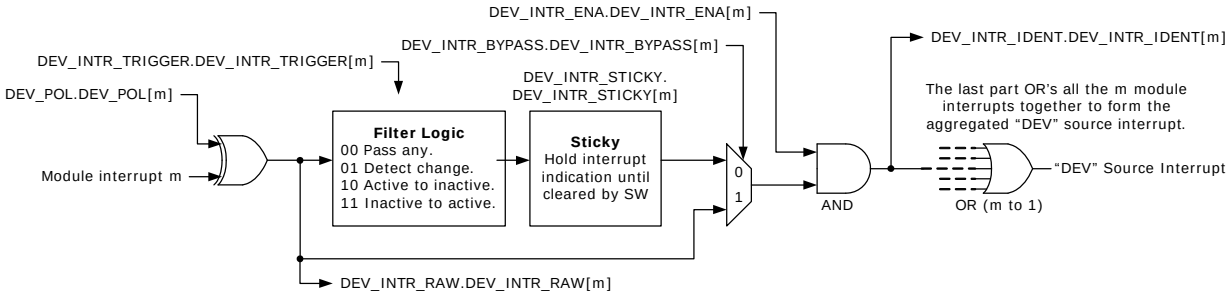
The interrupt destination can enable individual sources for interrupt by writing a mask to DST_INTR_MAP[d].DST_INTR_MAP. When a source is enabled in DST_INTR_MAP then an interrupt from this source will be propagated to the interrupt destination.

The currently active and enabled source interrupts for a destination can be seen by reading DST_INTR_IDENT[d].DST_INTR_IDENT.

4.8.13.3 Port Module Interrupts

Each port module can generate an interrupt. Because there are too many modules to handle the interrupts in parallel with the other source interrupts in the INTR_* registers, the port module interrupts are aggregated in a separate source interrupt hierarchy before being presented to the interrupt controller source logic as the DEV source interrupt.

***Figure 170* • Port Module Interrupt Logic**



The module interrupt polarity is configurable in DEV_INTR_POL.DEV_INTR_POL[m].

DEV_INTR_RAW, DEV_INTR_TRIGGER, DEV_INTR_STICKY, DEV_INTR_BYPASS, DEV_INTR_ENA, and DEV_INTR_IDENT works in the same way as the INTR_* counterparts.For more information, see Interrupt Source Configuration, page 454.

The final step when handling module interrupts is an aggregation of all individual module interrupts to the DEV source interrupt.

4.8.13.4 External Interrupts

The interrupt controller supports two external source interrupts (inputs to the device) and two external destination interrupts (outputs from the device). The external interrupts are mapped to GPIOs using overlaid functions. For more information about overlaid functions on the GPIOs for these signals, see GPIO Overlaid Functions, page 439.

Source and destination interrupts works independently from each other and can be used at the same time. The polarity (active high or low) of source and destination interrupts is configured in EXT_SRC_INTR_POL and EXT_DST_INTR_POL respectively.

Table 276 • **External Interrupt Pins**

Register	I/O	Description
IRQ0_IN/GPIO	I	External Source Interrupt 0. Polarity is configured in EXT_SRC_INTR_POL.EXT_INTR_POL[0].
IRQ1_IN/GPIO	I	External Source Interrupt 1. Polarity is configured in EXT_SRC_INTR_POL.EXT_INTR_POL[1].
IRQ0_OUT/GPIO	O	External Destination Interrupt 0. Polarity is configured in EXT_DST_INTR_POL.EXT_INTR_POL[0]. This interrupt can also be mapped to GPIO (replaces source interrupt).
IRQ1_OUT/GPIO	O	External Destination Interrupt 1. Polarity is configured in EXT_DST_INTR_POL.EXT_INTR_POL[1]. This interrupt can also be mapped to GPIO (replaces source interrupt).

For destination interrupts it is possible to drive the output pin permanently or emulate open-collector output.

- To drive permanently, configure EXT_INTR_DRV[e] = 0.
- To emulate open collector output, configure EXT_INTR_DRV[e] = 1 and EXT_INTR_POL[e] = 0. To safely enable open-collector output, the EXT_INTR_DRV and EXT_INTR_POL registers must be configured before enabling the overlaid function in the GPIO controller.

Note: Open collector output mode is required when multiple interrupt sources are hooked up to the same interrupt wire on the PCB and the wire is be pulled high with a resistor. Each interrupt source can then drive the wire low via open-collector output when they want to signal interrupt.

5 Registers

Information about the registers for this product is available in the attached Adobe Acrobat file. To view or print the information, double-click the attachment icon.

6 Electrical Specifications

This section provides the DC characteristics, AC characteristics, recommended operating conditions, and stress ratings for the VSC7430 device.

6.1 DC Characteristics

This section contains the DC specifications for the VSC7430 device.

6.1.1 Reference Clock

The following table lists the DC specifications for the differential reference clock signals. Differential and single-ended modes are supported.

Table 277 • **Reference Clock Inputs**

Parameter	Symbol	Minimum	Typical	Maximum	Unit
Input voltage range	V_{IP}, V_{IN}	–25		1200	mV
Single-ended input swing	$\|V_{SE}\|$	600		1000[1]	mV
Differential peak-to-peak input swing	$\|V_{ID}\|$	200		1200	mVppd
Input common-mode voltage	V_{CM}		$^2/_3 \times V_{DD}$		mV

1. Input common-mode voltage and amplitude must not exceed 1200 mV.

6.1.2 PLL Clock Output

The following table lists the DC specifications for the PLL clock outputs. Applies to CLKOUTPLL and CLKOUTPLL2 pins.

Table 278 • **PLL Clock Output**

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
Differential resistance	R_{DIFF}	80	100	120	Ω	
Differential peak-to-peak output swing	V_{OD}	290		510	mVppd	
Differential peak-to-peak output swing	V_{OD}	360		635	mVppd	See note [1].
Output common-mode voltage	V_{CM}	V_{DD_A} – 500 mV		V_{DD_A}	mV	

1. Driver amplitude depends on driver-to-receiver adaptation configured in register. Increased amplitude can be achieved if the receiver is common-mode terminated to V_{DD_A} and if the driver is configured accordingly.

6.1.3 DDR3/DDR3L SDRAM Interface

This section provides the DC specifications for the DDR3 and DDR3L interfaces.

The DDR3 SDRAM interface supports the requirements of SDRAM devices as described in the JEDEC DDR3 specifications. The SDRAM interface signals are compatible with JESD79-3F (DDR3 SDRAM SPECIFICATION, July 2012) and JESD79-3-1A, January 2013. The SSTL I/O buffers have programmable on-die termination (ODT).

6.1.3.1 DDR3 SDRAM Interface

The following table lists the DC specifications for the SDRAM interface signals in DDR3 operation.

Table 279 • **DDR3 SDRAM Signals**

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Input reference voltage[1]	DDR_V_{REF}	49% V_{DD_IODDR}	51% V_{DD_IODDR}	V	V_{DD_IODDR} = 1.5 V.
Input voltage high	$V_{IH(DC)}$	DDR_V_{REF} + 0.100	V_{DD_IODDR}	V	
Input voltage low	$V_{IL(DC)}$	–0.3	DDR_V_{REF} – 0.100	V	
Output voltage high	$V_{OH(DC)}$	0.8 × V_{DD_IODDR}		V	Not terminated, 1 pF load.
Output voltage low	$V_{OL(DC)}$	–0.3	0.2 × V_{DD_IODDR}	V	Not terminated, 1 pF load.
Input leakage current	$\|I_L\|$		40	µA	0 V ≤ V_I ≤ V_{DD_IODDR}.
Output source DC current[2]	I_{OH}	–6		mA	External 40 Ω termination to V_{DD_IODDR}/2.
Output sink DC current[2]	I_{OL}	6		mA	External 40 Ω termination to V_{DD_IODDR}/2.

1. DDR_V_{REF} is expected to track variations in V_{DD_IODDR}. Peak-to-peak AC noise on DDR_V_{REF} must not exceed ±2% of DDR_V_{REF}.
2. With 34 Ω output driver impedance.

6.1.3.2 DDR3L SDRAM Interface

The following table lists the DC specifications for the SDRAM interface signals in DDR3L operation.

Table 280 • **DDR3L SDRAM Signals**

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Input reference voltage[1]	DDR_V_{REF}	49% V_{DD_IODDR}	51% V_{DD_IODDR}	V	V_{DD_IODDR} = 1.35 V.
Input voltage high	$V_{IH(DC)}$	DDR_V_{REF} + 0.09	V_{DD_IODDR}	V	
Input voltage low	$V_{IL(DC)}$	–0.3	DDR_V_{REF} –0.09	V	
Output voltage high	$V_{OH(DC)}$	0.8 × V_{DD_IODDR}		V	Not terminated, 1 pF load.
Output voltage low	$V_{OL(DC)}$		0.2 × V_{DD_IODDR}	V	Not terminated, 1 pF load.
Input leakage current	$\|I_L\|$		40	µA	0 V ≤ V_I ≤ V_{DD_IODDR}.
Output source DC current[2]	I_{OH}	–6		mA	External 40 Ω termination to V_{DD_IODDR}/2
Output sink DC current[2]	I_{OL}	6		mA	External 40 Ω termination to V_{DD_IODDR}/2

1. DDR_V_{REF} is expected to track variations in V_{DD_IODDR}. Peak-to-peak AC noise on DDR_V_{REF} must not exceed ±2% of DDR_V_{REF}.
2. With 34 Ω output driver impedance.

6.1.4 SERDES1G

This section describes the DC specifications for the SERDES1G transceiver. The transceiver supports 100BASE-FX, SFP, 1000BASE-KX, and SGMII modes.

The following table lists the DC specifications for the 1G transmitter. Applies to the S[4:0]_TXN/P pins.

Table 281 • **1G Transmitter**

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
Differential resistance	R_{DIFF}	80	100	120	Ω	
Output voltage high	V_{OH}			1050	mV	
Output voltage low	V_{OL}	0			mV	
Differential peak-to-peak output voltage[1]	V_{O_DIFF}	300		800	mVppd	100BASE-FX, SGMII
Differential peak-to-peak output voltage[1]	V_{O_DIFF}	500		1200	mVppd	SFP
Differential peak-to-peak output voltage[1]	V_{O_DIFF}	800		1100	mVppd	1000BASE-KX
Differential peak-to-peak output voltage with Tx disabled	V_{OD_IDLE}			30	mVppd	

1. Output amplitude is configurable in 16 steps.

The following table lists the DC specifications for the 1G receiver. Applies to the S[4:0]_RXN/P pins.

Table 282 • **1G Receiver**

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
Differential resistance	R_{DIFF}	80	100	120	Ω	
Absolute input voltage range	V_{IN}	–25		1200	mV	
Common-mode voltage	V_{CM_AC}	0		V_{DD_A}	mV	AC-coupled operation[1]
Common-mode voltage	V_{CM_DC}		$0.7 \times V_{DD_A}$		mV	DC-coupled operation[2, 3]
Differential peak-to-peak input voltage	V_{IN_DIFF}	100		1600	mVppd	See note [4]

1. Compatibility to SGMII transmitters requires external AC-coupling. The maximum common-mode voltage is provided without a differential signal. It is limited by the minimum and maximum input voltage range and the input's signal amplitude.
2. For information about optional DC-coupling, contact your Microsemi sales representative.
3. Common-mode termination disabled. The maximum differential peak-to-peak input is limited by the maximum input voltage range.
4. Applies to all supported modes. For 100BASE-FX, disable internal AC-coupling.

6.1.5 SERDES6G

This section provides the DC specifications for the 6G transceiver. The transceiver supports the following modes:

- 100BASE-FX
- SGMII
- SFP
- PCIe
- 2.5G

The following table lists the DC specifications for the 6G transmitter. Applies to the PCIE_TXN/P and S[6:5]_TXN/P pins.

Table 283 • **6G Transmitter**

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
Differential resistance	R_{DIFF}	80	100	120	Ω	
Output voltage high	V_{OH}			V_{DD_VS}	mV	

Table 283 • **6G Transmitter *(continued)***

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
Output voltage low	V_{OL}	0			mV	
Differential peak-to-peak output voltage[1]	V_{O_DIFF}	300		800	mVppd	100BASE-FX, SGMII[2]
Differential peak-to-peak output voltage[1]	V_{O_DIFF}	500		1200	mVppd	SFP, 2.5G
Differential peak-to-peak output voltage[1]	V_{O_DIFF}	800		1200	mVppd	1000BASE-KX and PCIe[3]
Differential peak-to-peak output voltage with Tx disabled	V_{OD_IDLE}			30	mVppd	
Output current, driver shorted to GND	T_ISG			40	mA	

1. Drive level depends on register configuration.
2. Compatibility to SGMII receiver requires AC-coupling.
3. Compatibility to supported standards require 1.2 V supply for driver.

The following table lists the DC specifications for the 6G receiver. Applies to the PCIE_RXN/P and S[6:5]_RXN/P pins.

Table 284 • **6G Receiver**

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
Differential resistance	R_{DIFF}	80	100	120	Ω	
Absolute input voltage range	V_{IN}	–25		1200	mV	
Common-mode voltage	V_{CM}	0	Internal V_{CM}		mV	AC-coupled operation[1]
Common-mode voltage	V_{CM}		V_{DD_A}		mV	DC-coupled operation, load type 2[1]
Differential peak-to-peak input voltage	V_{IN_DIFF}	100		1600	mVppd	See note [2]

1. Mode for common-mode termination is specified by configuration register setting. Input amplitude in DC-coupled mode must not exceed maximum input voltage range. For more information about optional DC-coupling, contact your Microsemi representative.
2. Compatibility to SGMII transmitter requires AC-coupling.

6.1.6 GPIO, SI, JTAG, and Miscellaneous Signals

This section provides the DC specifications for the GPIO, SI, JTAG, and miscellaneous signals.

The following I/O signals comply with the specifications provided in this section:

Table 285 • **I/O Signals**

GPIO[36:0]	JTAG_nTRST	nRESET
SI_CLK	JTAG_TMS	
SI_DI	JTAG_TDO	
SI_DO	JTAG_TCK	
SI_nCS0	JTAG_TDI	
	JTAG_ICE_nEN	

The outputs and inputs meet or exceed the requirements of the LVTTL and LVCMOS standard, JEDEC JESD8-B (September 1999) standard, unless otherwise stated. The inputs are Schmitt-trigger for noise immunity.

Table 286 • **GPIO, SI, JTAG, and Miscellaneous Signals DC Specifications**

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
Output high voltage[1]	V_{OH}	2.1	2.35		V	I_{OH} = –2 mA
Output high voltage[1]	V_{OH}	1.7	2.0		V	I_{OH} = –12 mA
Output low voltage	V_{OL}			0.4	V	I_{OL} = 2mA
Output low voltage	V_{OL}			0.7	V	I_{OL} = 12 mA
Input high voltage	V_{IH}	1.85		3.6	V	
Input low voltage	V_{IL}	–0.3		0.8	V	
Input high current[2]	I_{IH}			10	µA	$V_I = V_{DD_IO}$
Input low current[2]	I_{IL}	–100			µA	V_I = 0 V
Input capacitance	C_I			10	pF	

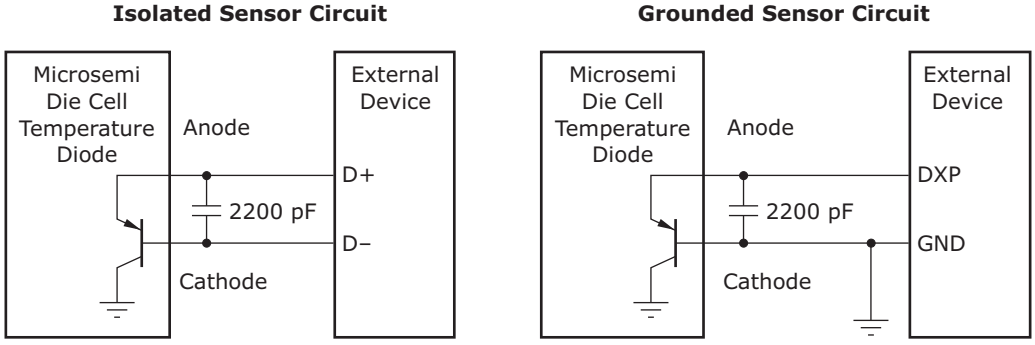
1. V_{DD_IO} = 2.38 V minimum, V_{DD_IO} = 2.5 typical.
2. Input high current and input low current equals the maximum leakage current, excluding the current in the built-in pull resistors.

6.1.7 Thermal Diode

The device includes an on-die diode and internal circuitry for monitoring die temperature (junction temperature). The operation and accuracy of the diode is not guaranteed and should only be used as a reference.

The on-die thermal diode requires an external thermal sensor, located on the board or in a stand-alone measurement kit. Temperature measurement using a thermal diode is very sensitive to noise. The following illustration shows a generic application design.

Figure 171 • **Thermal Diode**



Note: Microsemi does not support or recommend operation of the thermal diode under reverse bias.

The following table provides the diode parameter and interface specifications with the pins connected internally to VSS in the device.

Table 287 • **Thermal Diode Parameters**

Parameter	Symbol	Typical	Maximum	Unit
Forward bias current	I_{FW}	See note[1]	1	mA
Diode ideality factor	n	1.008		

1. Typical value is device dependent.

The ideality factor, n, represents the deviation from ideal diode behavior as exemplified by the following diode equation:

$$I_{FW} = I_S(e^{(qV_D)/(nkT)} - 1)$$

where, I_S = saturation current, q = electron charge, V_D = voltage across the diode, k = Boltzmann constant, and T = absolute temperature (Kelvin).

6.2 AC Characteristics

This section provides the AC specifications for the VSC7430 device.

6.2.1 Reference Clock

The signal applied to the REFCLK differential inputs must comply with the requirements listed in the following tables at the pin of the device.

The following table lists the AC specifications for the REFCLK reference clock.

Table 288 • **REFCLK Reference Clock**

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
REFCLK frequency REFCLK_CONF = 100	*f*	–100 ppm	25	100 ppm	MHz	
REFCLK frequency REFCLK_CONF = 000	*f*	–100 ppm	125	100 ppm	MHz	
REFCLK frequency REFCLK_CONF = 001	*f*	–100 ppm	156.25	100 ppm	MHz	
REFCLK frequency REFCLK_CONF = 010	*f*	–100 ppm	250	100 ppm	MHz	
Clock duty cycle		40		60	%	Measured at 50% threshold.
Rise time and fall time	t_R, t_F			0.5	ns	Within ±200 mV relative to $V_{DD} \times {}^{2}/_{3}$.
Jitter transfer from REFCLK to SerDes output, bandwidth from 10 kHz to 1 MHz				0.3	dB	
Jitter transfer from REFCLK to SerDes output, bandwidth from 1 MHz to 10 MHz				2	dB	125 MHz reference clock.
Jitter transfer from REFCLK to SerDes output, bandwidth above 10 MHz				2 – 20 × log (*f*/7 MHz)	dB	125 MHz reference clock.
REFCLK input peak-to-peak jitter, bandwidth from 2.5 kHz to 10 MHz[1]				20	ps	To meet G.8262 1G SyncE jitter generation specification.

1. Peak-to-peak values are typically higher than the RMS value by a factor of 10 to 14.

The following table lists the AC specifications for the REFCLK2 reference clock.

Table 289 • **REFCLK2 Reference Clock**

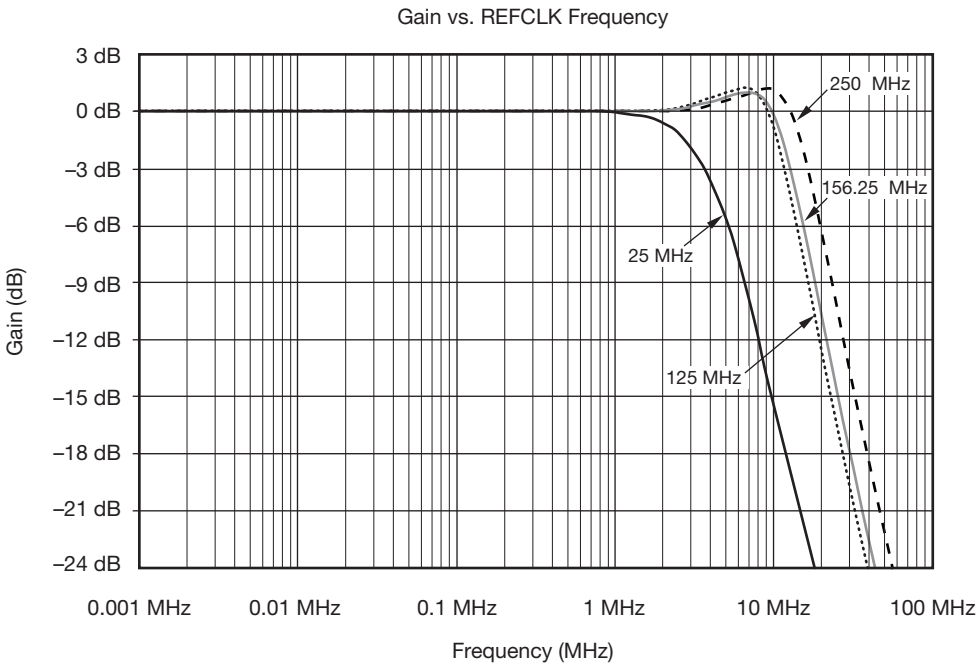
Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
REFCLK2 frequency REFCLK2_CONF = 100	*f*	–100 ppm	25	100 ppm	MHz	

Table 289 • **REFCLK2 Reference Clock *(continued)***

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
REFCLK2 frequency REFCLK2_CONF= 000	f	–100 ppm	125	100 ppm	MHz	
REFCLK2 frequency REFCLK2_CONF = 001	f	–100 ppm	156.25	100 ppm	MHz	
REFCLK2 frequency REFCLK2_CONF = 010	f	–100 ppm	250	100 ppm	MHz	
Clock duty cycle		40		60	%	Measured at 50% threshold.
Rise time and fall time, differential mode	t_R, t_F			0.5	ns	Within ±200 mV relative to $V_{DD} \times {}^{2}/_{3}$.
Rise time and fall time, 25 MHz single-ended mode	t_R, t_F			2	ns	Within ±200 mV relative to $V_{DD} \times {}^{2}/_{3}$.
Jitter transfer from REFCLK2 to SERDES10G/clock outputs, bandwidth from 300 kHz to 3 MHz				0.6	dB	
Jitter transfer from REFCLK2 to SERDES10G/clock outputs, bandwidth from 3 MHz to 12 MHz				2	dB	
Jitter transfer from REFCLK2 to SERDES10G/clock outputs, bandwidth above 12 MHz				2 – 20 × log (f/12 MHz)	dB	
REFCLK2 input RMS jitter, 12 kHz to 10 MHz				1	ps_{RMS}	

The following illustration shows jitter transfer curves from REFCLK to all high-speed outputs.

Figure 172 • **REFCLK Jitter Transfer Curves**



6.2.2 PLL Clock Outputs

The following table lists the AC specifications for the PLL outputs. Applies to the CLKOUTPLL and CLKOUTPLL2 pins.

Table 290 • **PLL Clock Outputs**

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Output frequency			625	MHz	
Clock duty cycle	t_C	45	55	%	Measured at 50% threshold
Rise time and fall time	t_R, t_F	100	300	ps	20% to 80% of V_S
Intrapair skew	t_{SKEW}		100	ps	
Jitter generation, 10 kHz to 50 MHz			4	ps_{RMS}	Jitter-free input used for REFCLK/REFCLK2

6.2.3 SERDES1G

This section describes the AC specifications for the SERDES1G transceiver. The transceiver supports 100BASE-FX, SFP, 1000BASE-KX, and SGMII modes.

The following table lists the AC characteristics for the 1G transmitter. Applies to S[4:0]_TXN/P pins.

Table 291 • **100BASE-FX, SGMII, SFP, 1000BASE-KX Transmitter**

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Data rate		125 – 100 ppm	125 + 100 ppm	Mbps	100BASE-FX.
Data rate		1.25 – 100 ppm	1.25 + 100 ppm	Gbps	SGMII, SFP, 1000BASE-KX.
Differential output return loss	RLO_{SDD22}		–10	dB	50 MHz to 625 MHz.
Differential output return loss	RLO_{SDD22}		–10 + 10 x log (*f*/625 MHz)	dB	625 MHz to 1250 MHz.
Rise time and fall time[1]	t_R, t_F	60	300	ps	20% to 80%.
Interpair skew	t_{SKEW}		20	ps	
Deterministic jitter	DJ		80	ps	Measured according to IEEE 802.3 Clause 38.5.
Total jitter	TJ		192	ps	Measured according to IEEE 802.3 Clause 38.5.
Wideband SyncE jitter	WJT		0.5	UI_{P-P}	Measured according to ITU-T G.8262, section 8.3.

1. Slew rate is programmable.

The following table lists AC characteristics for the 1G receiver. Applies to the S[4:0]_RXN/P pins.

Table 292 • **100BASE-FX, SGMII, SFP, 1000BASE-KX Receiver**

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Data rate		125 – 100 ppm	125 + 100 ppm	Mbps	100BASE-FX.
Data rate		1.25 – 100 ppm	1.25 + 100 ppm	Gbps	SGMII, SFP, 1000BASE-KX.
Differential input return loss	RLI_{SDD11}		–10	dB	50 MHz to 625 MHz.

Table 292 • **100BASE-FX, SGMII, SFP, 1000BASE-KX Receiver *(continued)***

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Differential input return loss	RLI_{SDD11}		–10 + 10 x log (*f*/625 MHz)	dB	625 MHz to 1250 MHz.
Jitter tolerance, total[1]	TOL_{TJ}	600		ps	SGMII, SFP, 1000BASE-KX. Measured according to IEEE 802.3 Clause 38.6.8.
Jitter tolerance, deterministic[1]	TOL_{DJ}	370		ps	SGMII, SFP, 1000BASE-KX. Measured according to IEEE 802.3 Clause 38.6.8.
Jitter tolerance, duty cycle distortion	TOL_{DCD}	1.4		ns_{P-P}	100BASE-FX. Measured according to ISO/IEC 9314-3:1990.
Jitter tolerance, data dependent	TOL_{DDJ}	2.2		ns_{P-P}	100BASE-FX. Measured according to ISO/IEC 9314-3:1990.
Jitter tolerance, random	TOL_{RJ}	2.27		ns_{P-P}	100BASE-FX. Measured according to ISO/IEC 9314-3:1990.
Wideband SyncE jitter tolerance	WJT	312.5		UI_{P-P}	10 Hz to 12.1 Hz. Measured according to ITU-T G.8262, section 9.2.
Wideband SyncE jitter tolerance	WJT	3750/*f*		UI_{P-P}	12.1 Hz to 2.5 kHz (*f*) Measured according to ITU-T G.8262, section 9.2.
Wideband SyncE jitter tolerance	WJT	1.5		UI_{P-P}	2.5 kHz to 50 kHz. Measured according to ITU-T G.8262, section 9.2.

1. Jitter requirements represent high-frequency jitter (above 637 kHz) and not low-frequency jitter or wander.

6.2.4 SERDES6G

This section describes the AC specifications for the 6G transceiver. The transceiver supports the following modes:

- 100BASE-FX
- SGMII
- SFP
- 2.5G
- PCIe
- 1000BASE-KX

The following table lists the AC characteristics for the 6G transmitter in 100BASE-FX, SGMII, SFP, 2.5G, and 1000BASE-KX modes. Applies to the S[6:5]_TXN/P pins.

Table 293 • **100BASE-FX, SGMII, SFP, 2.5G, 1000BASE-KX Transmitter**

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Data rate		125 – 100 ppm	125 + 100 ppm	Mbps	100BASE-FX
Data rate		1.25 – 100 ppm	1.25 + 100 ppm	Gbps	SGMII, SFP, 1000BASE-KX
Data rate		3.125 – 100 ppm	3.125 + 100 ppm	Gbps	2.5G

Table 293 • **100BASE-FX, SGMII, SFP, 2.5G, 1000BASE-KX Transmitter *(continued)***

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Differential output return loss	RLO_{SDD22}		–10	dB	50 MHz to 625 MHz
Differential output return loss	RLO_{SDD22}		–10 + 10 x log (*f*/625 MHz)	dB	625 MHz to 1250 MHz
Rise time and fall time[1]	t_R, t_F	60	320	ps	20% to 80%
Interpair skew	t_{SKEW}		20	ps	
Random jitter	RJ		0.15	UI_{P-P}	At BER 10^{-12}
Deterministic jitter	DJ		0.10	UI_{P-P}	
Total jitter	TJ		0.25	UI_{P-P}	
Wideband SyncE jitter	TWJ		0.5	UI_{P-P}	Measured according to ITU-T G.8262, section 8.3
Eye mask	X1		0.125	UI	
Eye mask	X2		0.325	UI	
Eye mask	Y1	350[2]		mV	
Eye mask	Y2		800	mV	

1. Slew rate is programmable.
2. Compatibility to supported standard requires V_{DD_VS} = 1.2 V.

The following table lists AC characteristics for the 6G receiver operating in 100BASE-FX, SGMII, SFP, 2.5G, and 1000BASE-KX modes. Applies to the S[6:5]_RXN/P pins.

Table 294 • **100BASE-FX, SGMII, SFP, 2.5G, 1000BASE-KX Receiver**

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Data rate		125 – 100 ppm	125 + 100 ppm	Mbps	100BASE-FX.
Data rate		1.25 – 100 ppm	1.25 + 100 ppm	Gbps	SGMII, SFP, 1000BASE-KX.
Data rate		3.125 – 100 ppm	3.125 + 100 ppm	Gbps	2.5G.
Differential input return loss	RLI_{SDD11}		–10	dB	50 MHz to 625 MHz.
Differential input return loss	RLI_{SDD11}		–10 + 10 x log (*f*/625 MHz)	dB	625 MHz to 1250 MHz.
Jitter tolerance, total[1]	TOL_{TJ}	600		ps	Measured according to IEEE 802.3 Clause 38.6.8.
Jitter tolerance, deterministic[1]	TOL_{DJ}	370		ps	Measured according to IEEE 802.3 Clause 38.6.8.
Jitter tolerance, duty cycle distortion	TOL_{DCD}	1.4		ns_{P-P}	100BASE-FX. Measured according to ISO/IEC 9314-3:1990.
Jitter tolerance, data dependent	TOL_{DDJ}	2.2		ns_{P-P}	100BASE-FX. Measured according to ISO/IEC 9314-3:1990.
Jitter tolerance, random	TOL_{RJ}	2.27		ns_{P-P}	100BASE-FX. Measured according to ISO/IEC 9314-3:1990.

Table 294 • **100BASE-FX, SGMII, SFP, 2.5G, 1000BASE-KX Receiver *(continued)***

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Wideband SyncE jitter tolerance	WJT	312.5		UI_{P-P}	10 Hz to 12.1 Hz. Measured according to ITU-T G.8262, section 9.2.
Wideband SyncE jitter tolerance	WJT	3750/*f*		UI_{P-P}	12.1 Hz to 2.5 kHz (*f*). Measured according to ITU-T G.8262, section 9.2.
Wideband SyncE jitter tolerance	WJT	1.5		UI_{P-P}	2.5 kHz to 50 kHz. Measured according to ITU-T G.8262, section 9.2.

1. Jitter requirements represent high-frequency jitter (above 637 kHz) and not low-frequency jitter or wander.

The following table lists the AC characteristics for the 6G transmitter operating in PCIe mode. Applies to the PCIE_TXN/P pins.

Table 295 • **PCIe Transmitter**

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
Data rate		2.5 – 300 ppm		2.5 + 300 ppm	Gbps	
Differential output return loss	RLO_{SDD22}			–10	dB	50 MHz to 1.25 GHz
Common-mode return loss	RLO_{SCC22}			–6	dB	50 MHz to 1.25 GHz
De-emphasized differential output voltage (ratio)	Tode	–3	–3.5	–4	dB	
Rise time and fall time[1]	t_R, t_F	60			ps	20% to 80%. Recommended value.
Total jitter	TJ			0.25	UI_{P-P}	
Differential amplitude	$V_{TX_DIFF_PP}$	800[2]		1200	mV_{P-P}	

1. Slew rate is programmable. Configure accordingly for compliance to supported standard.
2. Compatibility to supported standard requires V_{DD_VS} = 1.2 V supply for driver.

The following table lists the AC characteristics for the 6G receiver operating in PCIe mode. Applies to the PCIE_RXN/P pins.

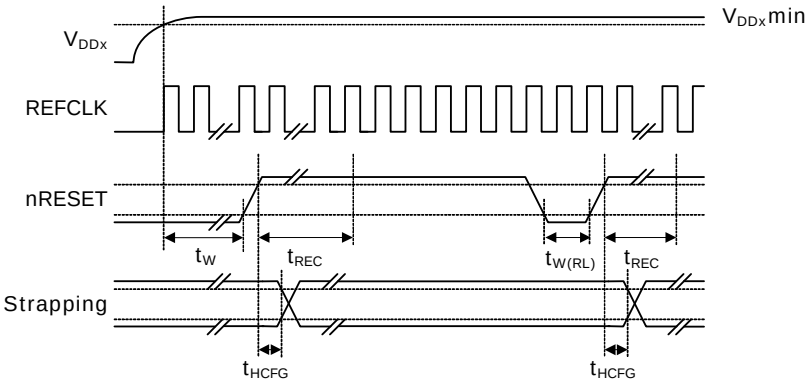
Table 296 • **PCIe Receiver**

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Data rate		2.5 – 300 ppm	2.5 + 300 ppm	Gbps	
Differential input return loss	RLI_{SDD11}		–10	dB	50 MHz to 1.25 GHz.
Common-mode return loss	RLI_{SCC11}		–6	dB	50 MHz to 2.5 GHz.
Total jitter tolerance	TJ		0.60	UI_{P-P}	
Eye mask	R_X1		0.30	UI	
Eye mask	R_X2		0.70	UI	
Eye mask	R_Y1	85		mV	
Eye mask	R_Y2		600	mV	

6.2.5 Reset Timing

The nRESET signal waveform and the required measurement points for the timing specification are shown in the following illustration.

Figure 173 • **nRESET Signal Timing Specifications**



The signal applied to the nRESET input must comply with the specifications listed in the following table at the reset pin of the device.

Table 297 • **nRESET Timing Specifications**

Parameter	Symbol	Minimum	Maximum	Unit
nRESET assertion time after power supplies and clock stabilizes	t_W	2		ms
Recovery time from reset inactive to device fully active	t_{REC}		10	ms
nRESET pulse width	$t_{W(RL)}$	100		ns
Hold time for GPIO-mapped strapping pins relative to nRESET	t_{HCFG}	50		ns

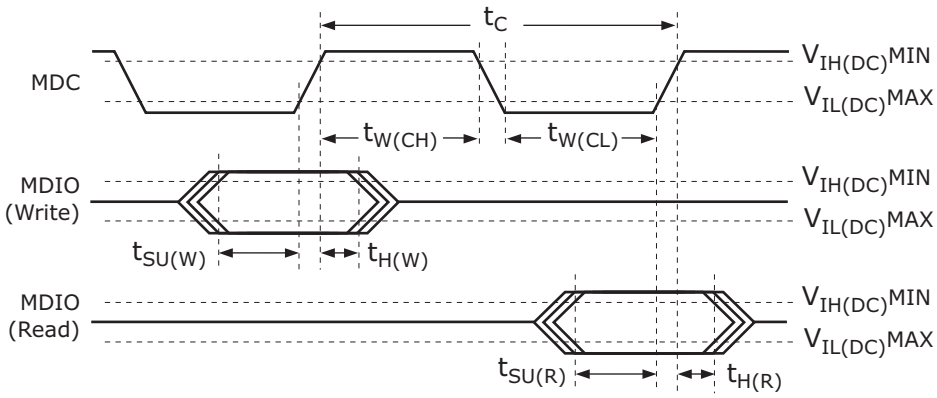
6.2.6 MII Management

All AC specifications for the MII Management (MIIM) interface meet or exceed the requirements of IEEE 802.3-2002 (clause 22.2-4).

All MIIM AC timing requirements are specified relative to the input low and input high threshold levels.

The following illustration shows the MIIM waveforms and required measurement points for the signals.

Figure 174 • **MIIM Timing Diagram**



The set up time of MDIO relative to the rising edge of MDC is defined as the length of time between when the MDIO exits and remains out of the switching region and when MDC enters the switching region. The

hold time of MDIO relative to the rising edge of MDC is defined as the length of time between when MDC exits the switching region and when MDIO enters the switching region.

The MIIM signals are GPIO alternate functions. For more information, see GPIO Overlaid Functions, page 439. All MIIM signals comply with the specifications in the following table. The MDIO signal requirements are requested at the pin of the device.

Table 298 • **MIIM Timing Specifications**

Parameter	Symbol	Minimum	Maximum	Unit	Condition
MDC frequency[1]	f	0.488	20.83	MHz	
MDC cycle time[2]	t_C	48	2048	ns	
MDC time high	$t_{W(CH)}$	20		ns	C_L = 50 pF
MDC time low	$t_{W(CL)}$	20		ns	C_L = 50 pF
MDIO setup time to MDC on write	$t_{SU(W)}$	15		ns	C_L = 50 pF
MDIO hold time from MDC on write	$t_{H(W)}$	15		ns	C_L = 50 pF
MDIO setup time to MDC on read	$t_{SU(R)}$	30		ns	C_L = 50 pF on MDC
MDIO hold time from MDC on read	$t_{H(R)}$	0		ns	C_L = 50 pF

1. For the maximum value, the device supports an MDC clock speed of up to 20 MHz for faster communication with the PHYs. If the standard frequency of 2.5 MHz is used, the MIIM interface is designed to meet or exceed the IEEE 802.3 requirements of the minimum MDC high and low times of 160 ns and an MDC cycle time of minimum 400 ns, which is not possible at faster speeds.
2. Calculated as $t_C = 1/f$.

6.2.7 Serial Interface (SI) Boot Master Mode

The following table lists the timing specifications for SI boot master mode.

Table 299 • **SI Boot Timing Specifications for Master Mode**

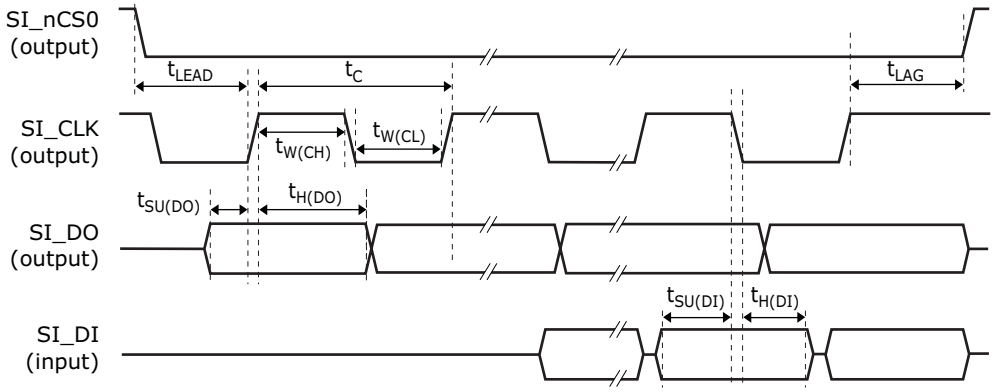
Parameter	Symbol	Minimum	Maximum	Unit	Condition
Clock frequency	f		25[1]	MHz	
Clock cycle time	t_C	40		ns	
Clock time high	$t_{W(CH)}$	16		ns	
Clock time low	$t_{W(CL)}$	16		ns	
Clock rise time and fall time	t_R, t_F		10	ns	Between $V_{IL(MAX)}$ and $V_{IH(MIN)}$. C_L= 30 pF.
SI_DO setup time to clock	$t_{SU(DO)}$	10		ns	
SI_DO hold time from clock	$t_{H(DO)}$	10		ns	
Enable active before first clock	t_{LEAD}	10		ns	
Enable inactive after clock	t_{LAG}	5		ns	
SI_DI setup time to clock	$t_{SU(DI)}$	22		ns	
SI_DI hold time from clock	$t_{H(DI)}$	–2		ns	

1. Frequency is programmable. The startup frequency is 8.1 MHz.

6.2.8 Serial Interface (SI) Master Mode

All serial interface (SI) timing requirements for master mode are specified relative to the input low and input high threshold levels. The following illustration shows the timing parameters and measurement points.

Figure 175 • **SI Timing Diagram for Master Mode**



All SI signals comply with the specifications shown in the following table. The SI input timing requirements are requested at the pins of the device.

Table 300 • **SI Timing Specifications for Master Mode**

Parameter	Symbol	Minimum	Maximum	Unit	Condition
Clock frequency	f		25[1]	MHz	
Clock cycle time	t_C	40		ns	
Clock time high	$t_{W(CH)}$	16		ns	
Clock time low	$t_{W(CL)}$	16		ns	
Clock rise time and fall time	t_R, t_F		10	ns	Between $V_{IL(MAX)}$ and $V_{IH(MIN)}$. C_L= 30 pF.
SI_DO setup time to clock	$t_{SU(DO)}$	10		ns	
SI_DO hold time from clock	$t_{H(DO)}$	10		ns	
Enable active before first clock	t_{LEAD}	10		ns	
Enable inactive after clock	t_{LAG}	15		ns	
SI_DI setup time to clock	$t_{SU(DI)}$	15		ns	
SI_DI hold time from clock	$t_{H(DI)}$	0		ns	

1. Frequency is programmable. The startup frequency is 4 MHz.

6.2.9 Serial Interface (SI) for Slave Mode

All serial interface (SI) slave mode timing requirements are specified relative to the input low and input high threshold levels. The following illustrations show the timing parameters and measurement points for SI input and output data.

***Figure 176* • SI Input Data Timing Diagram for Slave Mode**

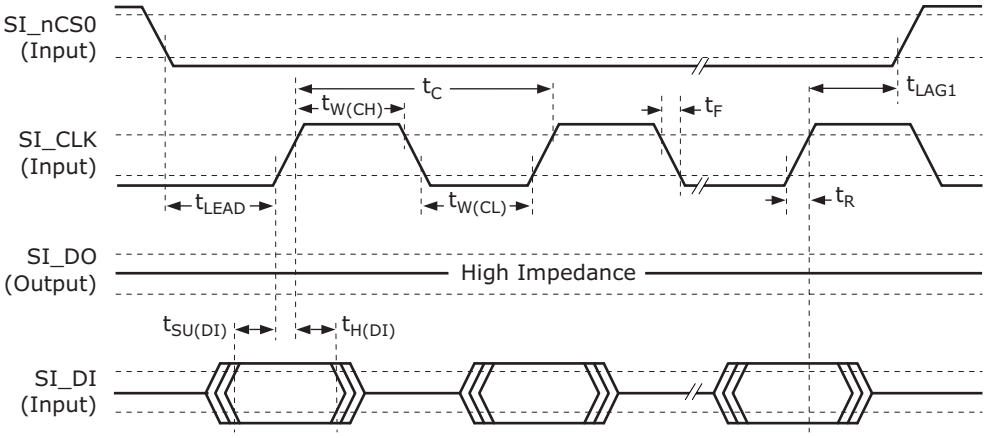
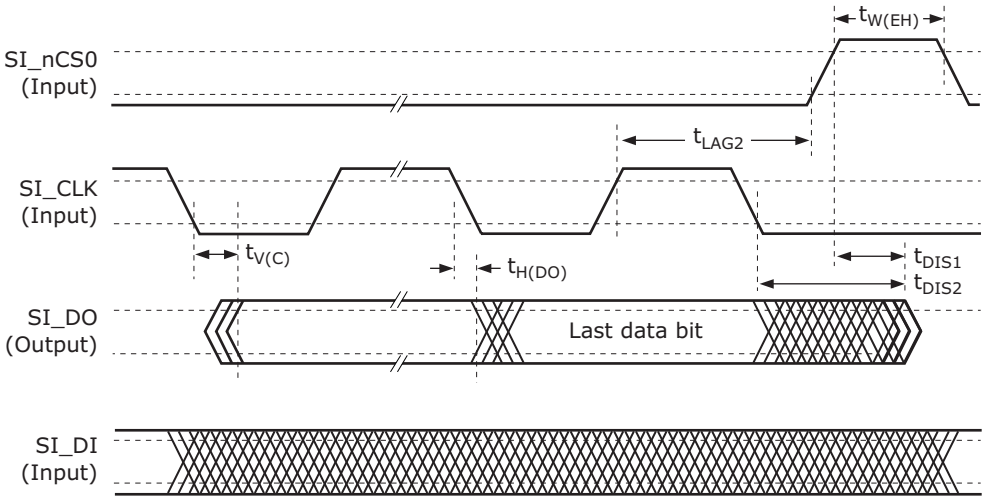


***Figure 177* • SI Output Data Timing Diagram for Slave Mode**



All SI signals comply with the specifications shown in the following table. The SI input timing requirements are requested at the pins of the device.

***Table 301* • SI Timing Specifications for Slave Mode**

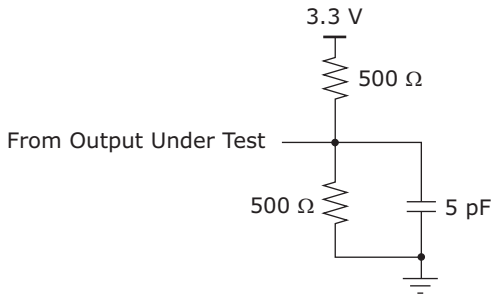
Parameter	Symbol	Minimum	Maximum	Unit	Condition
Clock frequency	f		25	MHz	
Clock cycle time	t_C	40		ns	
Clock time high	$t_{W(CH)}$	16		ns	
Clock time low	$t_{W(CL)}$	16		ns	
SI_DI setup time to clock	$t_{SU(DI)}$	4		ns	
SI_DI hold time from clock	$t_{H(DI)}$	4		ns	
Enable active before first clock	t_{LEAD}	10		ns	
Enable inactive after clock (input cycle)[1]	t_{LAG1}	25		ns	
Enable inactive after clock (output cycle)	t_{LAG2}	See note[2]		ns	
Enable inactive width	$t_{W(EH)}$	20		ns	
SI_DO valid after clock	$t_{V(C)}$		25	ns	C_L = 30 pF.
SI_DO hold time from clock	$t_{H(DO)}$	0		ns	C_L = 0 pF.

Table 301 • **SI Timing Specifications for Slave Mode *(continued)***

Parameter	Symbol	Minimum	Maximum	Unit	Condition
SI_DO disable time[3]	t_{DIS1}		20	ns	See Figure 177, page 472.
SI_DO disable time[3]	t_{DIS2}		20	ns	See Figure 177, page 472.

1. t_{LAG1} is defined only for write operations to the device, not for read operations.
2. The last rising edge on the clock is necessary for the external master to read in the data. The lag time depends on the necessary hold time on the external master data input.
3. Pin begins to float when a 300 mV change from the loaded V_{OH} or V_{OL} level occurs.

Figure 178 • **SI_DO Disable Test Circuit**

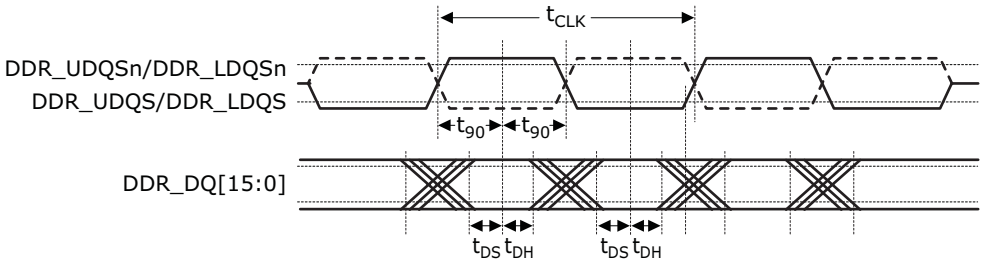


6.2.10 DDR SDRAM Interface

This section provides the AC characteristics for the DDR3 and DDR3L SDRAM interface.

The following illustration shows the DDR3/DDR3L SDRAM input timing diagram.

Figure 179 • **DDR SDRAM Input Timing Diagram**



The following table lists the AC specifications for the DDR3 and DDR3L SDRAM input signals.

Table 302 • **DDR3/DDR3L SDRAM Input Signal AC Characteristics**

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
DDR3 input voltage high	$V_{IH(AC)}$	DDR_V_{REF} + 0.175		V_{DD_IODDR} + 0.3	V	
DDR3 input voltage low	$V_{IL(AC)}$	–0.3		DDR_V_{REF} – 0.175	V	
DDR3 differential input voltage	$V_{ID(AC)}$	0.4		V_{DD_IODDR}	V	
DDR3 differential crosspoint voltage	$V_{IX(AC)}$	0.5 × V_{DD_IODDR} – 0.175		0.5 × V_{DD_IODDR} + 0.175	V	
DDR3L input voltage high	$V_{IH(AC)}$	DDR_V_{REF} + 0.160		V_{DD_IODDR} + 0.3	V	
DDR3L input voltage low	$V_{IL(AC)}$	–0.3		DDR_V_{REF} – 0.160	V	
DDR3L differential input voltage	$V_{ID(AC)}$	0.4		V_{DD_IODDR}	V	

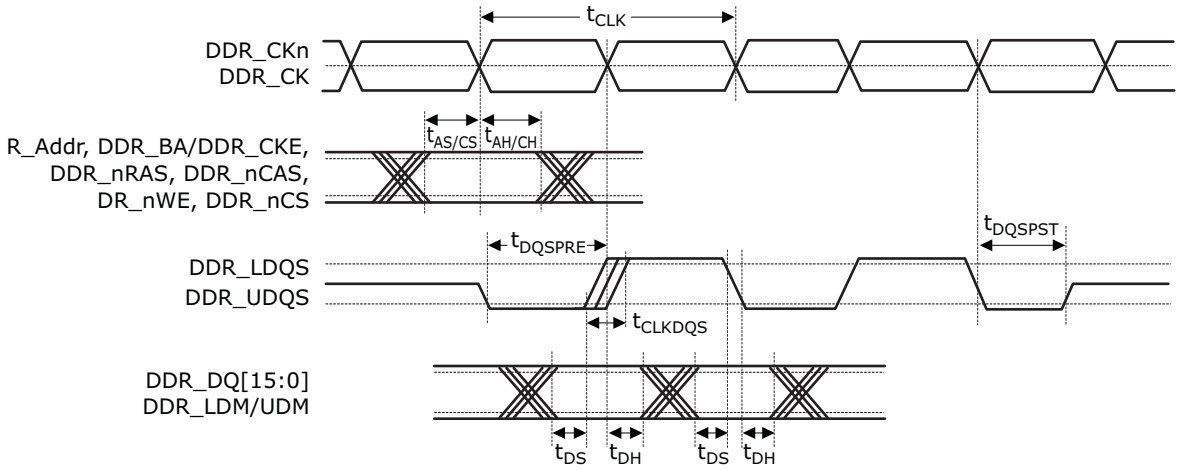
Table 302 • **DDR3/DDR3L SDRAM Input Signal AC Characteristics *(continued)***

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
DDR3L differential crosspoint voltage	$V_{IX(AC)}$	0.5 × V_{DD_IODDR} – 0.160		0.5 × V_{DD_IODDR} + 0.160	V	
DDR_DQ[7:0] input setup time relative to DDR_LDQS[1]	t_{DS}	350			ps	
DDR_DQ[15:8] input setup time relative to DDR_UDQS[1]	t_{DS}	350			ps	
DDR_DQ[7:0] input hold time relative to DDR_LDQS[1]	t_{DH}	250			ps	
DDR_DQ[15:8] input hold time relative to DDR_UDQS[1]	t_{DH}	250			ps	
Quarter period offset from clock edge	t_{90}		0.25 × t_{CLK}			t_{CLK} = 3.2 ns

1. These requirements are dependent on the operation frequency of the DDR SDRAM interface. Stated limits are for DDR3-800.

The following illustration shows the timing diagram for the DDR3 and DDR3L SDRAM outputs.

Figure 180 • **DDR SDRAM Output Timing Diagram**



The following table lists the AC characteristics for the DDR3 and DDR3L SDRAM output signals.

Table 303 • **DDR3/DDR3L SDRAM Output Signal AC Characteristics**

Parameter	Symbol	Minimum	Typical	Maximum	Unit
DDR_CK cycle time 312.5 MHz (DDR800)[1]	t_{CLK}		3.20		ns
DDR_CK/CKn duty cycle		48		52	%
DDR_A, DDR_BA, DDR_CKE, DDR_nRAS, DDR_nCAS, and DDR_nWE output setup time relative to DDR_CK/CKn	t_{AS}	700			ps

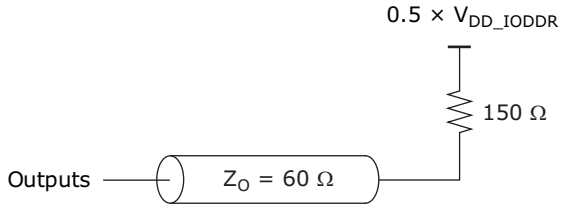
Table 303 • **DDR3/DDR3L SDRAM Output Signal AC Characteristics *(continued)***

Parameter	Symbol	Minimum	Typical	Maximum	Unit
DDR_A, DDR_BA, DDR_CKE, DDR_nRAS, DDR_nCAS, and DDR_nWE output hold time relative to DDR_CK/CKn	t_{AH}	800			ps
DDR_CK/CKn to DDR_DQS skew	t_{CLKDQS}	–400		400	ps
DDR_DQ[7:0]/DDR_LDM output setup time relative to DDR_LDQS	t_{DS}	400			ps
DDR_DQ[15:8]/DDR_UDM output setup time relative to DDR_UDQS	t_{DS}	400			ps
DDR_DQ[7:0]/DDR_LDM output hold time relative to DDR_LDQS	t_{DH}	500			ps
DDR_DQ[15:8]/DDR_UDM output hold time relative to DDR_UDQS	t_{DH}	500			ps
DDR_DQS preamble start	t_{DQSPRE}	0.4 × t_{CLK}		0.6 × t_{CLK}	ps
DDR_DQS postamble end	t_{DQSPST}	0.4 × t_{CLK}		0.6 × t_{CLK}	ps

1. Timing reference is DDR_CK/DDR_CKn crossing ±0.1 V.

The following illustration shows the test load circuit for the DDR3 outputs.

Figure 181 • **Test Load Circuit for DDR3 Outputs**

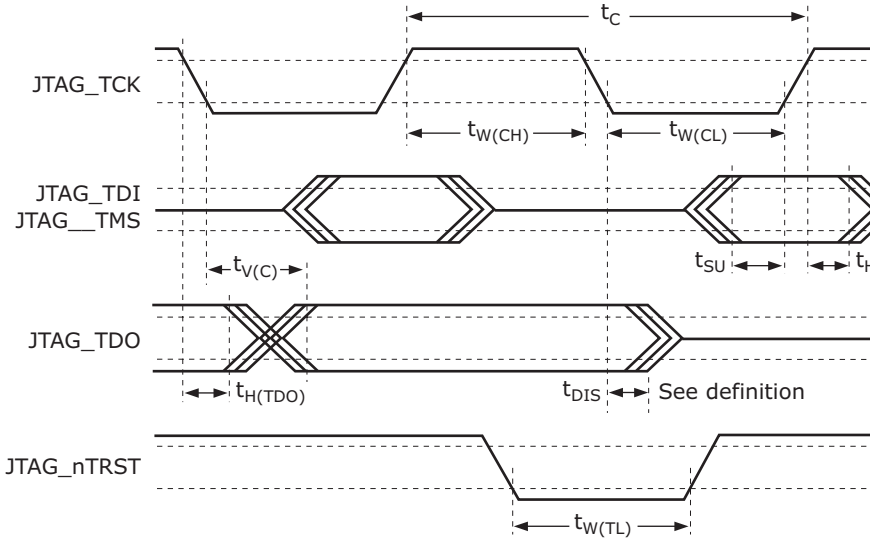


6.2.11 JTAG Interface

All AC specifications for the JTAG interface meet or exceed the requirements of IEEE 1149.1-2001.

The following illustration shows the JTAG transmit and receive waveforms and required measurement points for the different signals.

Figure 182 • **JTAG Interface Timing Diagram**



All JTAG signals comply with the specifications in the following table. The JTAG receive signal requirements are requested at the pin of the device.

The JTAG_nTRST signal is asynchronous to the clock and does not have a setup or hold time requirement.

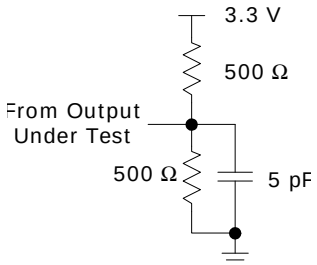
Table 304 • **JTAG Interface AC Specifications**

Parameter	Symbol	Minimum	Maximum	Unit	Condition
JTAG_TCK frequency	f		10	MHz	
JTAG_TCK cycle time	t_C	100		ns	
JTAG_TCK high time	$t_{W(CH)}$	40		ns	
JTAG_TCK low time	$t_{W(CL)}$	40		ns	
Setup time to JTAG_TCK rising	t_{SU}	10		ns	
Hold time from JTAG_TCK rising	t_H	10		ns	
TDO valid after JTAG_TCK falling	$t_{V(C)}$		28	ns	C_L = 10 pF
TDO hold time from JTAG_TCK falling	$t_{H(TDO)}$	0		ns	C_L = 0 pF
JTAG_TDO disable time[1]	t_{DIS}		30	ns	See Figure 183, page 476.
JTAG_nTRST time low	$t_{W(TL)}$	30		ns	

1. The pin begins to float when a 300 mV change from the actual V_{OH}/V_{OL} level occurs.

The following illustration shows the test circuit for the TDO disable time.

Figure 183 • **Test Circuit for TDO Disable Time**

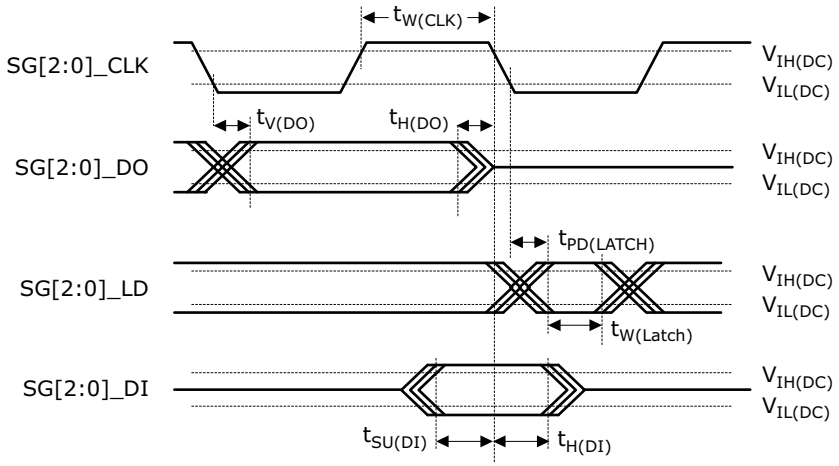


6.2.12 Serial Inputs/Outputs

This section provides the AC characteristics for the serial I/O signals: SG0_CLK, SG0_DO, SG0_DI, and SG0_LD. These signals are GPIO alternate functions. For more information, see GPIO Overlaid Functions, page 439.

The serial I/O timing diagram is shown in the following illustration.

Figure 184 • **Serial I/O Timing Diagram**



The following table lists the serial I/O timing specifications. Applies to the GPIO_[3:0] pins when used as serial I/O.

Table 305 • **Serial I/O Timing Specifications**

Parameter	Symbol	Minimum	Maximum	Unit
Clock frequency[1]	f		25	MHz
SG0_CLK clock pulse width	$t_{W(CLK)}$	40	60	%
SG0_DO valid after clock falling	$t_{V(DO)}$		6	ns
SG0_DO hold time from clock falling	$t_{H(DO)}$		6	ns
SG0_LD propagation delay from clock falling	$t_{PD(LATCH)}$	40		ns
SG0_LD width	$t_{W(LATCH)}$	10		ns
SG0_DI setup time to clock	$t_{SU(DI)}$	25		ns
SG0_DI hold time from clock	$t_{H(DI)}$	4		ns

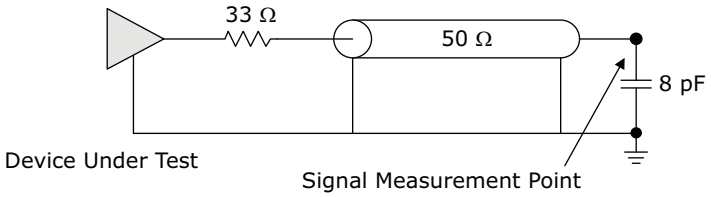
1. The SIO clock frequency is programmable.

6.2.13 Recovered Clock Outputs

This section provides the AC characteristics for the recovered clock output signals, RCVRD_CLK[3:0]. These signals are GPIO alternate functions. For more information about the GPIO pin mapping, see GPIO Overlaid Functions, page 439.

The following illustration shows the test circuit for the recovered clock output signals.

Figure 185 • **Test Circuit for Recovered Clock Output Signals**



The following table lists the AC specifications for the recovered clock outputs.

Table 306 • **Recovered Clock Output AC Specifications**

Parameter	Symbol	Minimum	Maximum	Unit	Condition
RCVRD_CLK[3:0] clock frequency	f		161.33	MHz	
Clock duty cycle	t_C	40	60	%	Measured at 50% threshold.
RCVRD_CLK[3:0] rise time and fall time	t_R, t_F		1.5	ns	
Squelching delay from SGMII signal to RCVRD_CLK[3:0]			200	ns	Squelch enabled.
RCVRD_CLK[3:0] peak-to-peak jitter, bandwidth between 12 kHz and 10 MHz[1], 60 second gate time			200	ps	Jitter-free input to SerDes Rx.
RCVRD_CLK[3:0] peak-to-peak jitter, bandwidth between 10 MHz and 80 MHz[1], 60 second gate time			200	ps	Jitter-free input to SerDes Rx.

1. Maximum jitter on the recovered signal.

6.2.14 Two-Wire Serial Interface

This section provides the AC specifications for the two-wire serial interface signals TWI_SCL and TWI_SDA. The two-wire serial interface signals are GPIO alternate functions. For more information about the GPIO pin mapping, see GPIO Overlaid Functions, page 439.

The two-wire serial interface signals are compatible with the Philips I2C-BUS specifications, except for the minimum rise time and fall time requirements for fast mode.

Figure 186 • **Two-Wire Serial Read Timing Diagram**

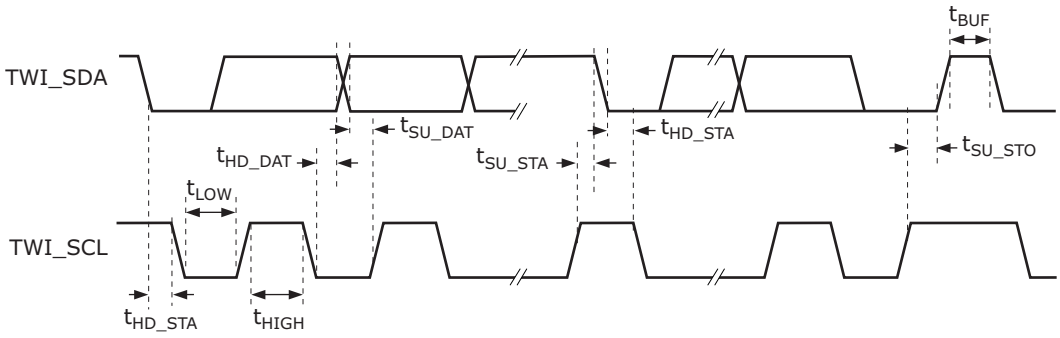
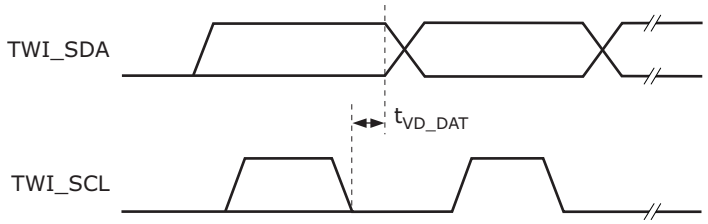


Figure 187 • **Two-Wire Serial Write Timing Diagram**



The following table lists the AC specifications for the serial interface. Standard mode is defined as 100 kHz and fast mode is 400 kHz. The data in this table assumes that the software-configurable two-wire interface timing parameters, SS_SCL_HCNT, SS_SCL_LCNT, FS_SCL_HCNT, and FS_SCL_LCNT, are set to valid values for the selected speed.

Table 307 • **Two-Wire Serial Interface AC Specifications**

		Standard Mode		Fast Mode			
Parameter	**Symbol**	**Minimum**	**Maximum**	**Minimum**	**Maximum**	**Unit**	**Condition**
TWI_SCL clock frequency	*f*		100		400	kHz	
TWI_SCL low period	t_{LOW}	4.7		1.3		µs	
TWI_SCL high period	t_{HIGH}	4.0		0.6		µs	
TWI_SCL and TWI_SDA rise time			1000		300	ns	
TWI_SCL and TWI_SDA fall time			300		300	ns	
TWI_SDA setup time to TWI_SCL fall	t_{SU_DAT}	250		100	300	ns	
TWI_SDA hold time to TWI_SCL fall[1]	t_{HD_DAT}	300	3450	300	900	ns	300 ns delay enabled in ICPU_CFG::TWI_CONFIG register.
Setup time for repeated START condition	t_{SU_STA}	4.7		0.6		µs	
Hold time after repeated START condition	t_{HD_STA}	4.0		0.6		µs	
Bus free time between STOP and START conditions	t_{BUF}	4.7		1.3		µs	
Clock to valid data out[2]	t_{VD_DAT}	300		300		ns	
Pulse width of spike suppressed by input filter on TWI_SCL or TWI_SDA		0	5	0	5	ns	

1. An external device must provide a hold time of at least 300 ns for the TWI_SDA signal to bridge the undefined region of the falling edge of the TWI_SCL signal.
2. Some external devices may require more data in hold time (target device's t_{HD_DAT}) than what is provided by t_{VD_DAT}, for example, 300 ns to 900 ns. The minimum value of t_{VD_DAT} is adjustable; the value given represents the recommended minimum value, which is enabled in ICPU_CFG::TWI_CONFIG.TWI_DELAY_ENABLE.

6.2.15 IEEE 1588 Time Tick Outputs

This section provides the AC specifications for the IEEE 1588 time tick output signals PTP_[3:0]. The PTP signals are GPIO alternate functions. For more information, see GPIO Overlaid Functions, page 439.

Table 308 • **IEEE1588 Time Tick Output AC Specifications**

Parameter	Symbol	Minimum	Maximum	Unit	Condition
PTP[3:0] frequency[1]	*f*		25	MHz	
Clock duty cycle[2]		45	55	%	Measured at 50% threshold.
PTP[3:0] rise time and fall time	t_R, t_F	1		ns	20% to 80% threshold.
PTP[3:0] peak-to-peak jitter, bandwidth between 12 kHz and 10 MHz[3]			200	ps	
PTP[3:0] peak-to-peak jitter, bandwidth between 10 MHz and 80 MHz[3]			200	ps	

1. Frequency is programmable.
2. Both high and low clock periods are configured to the identical value using DEVCPU_PTP::PIN_WF_HIGH_PERIOD and DEVCPU_PTP::PIN_WF_LOW_PERIOD.
3. Timing corrections performed by register writes and high/low periods that do not match a fixed number of 156.25 MHz clock cycles will create jitter up to 3.5 ns.

6.3 Current and Power Consumption

This section provides the current and power consumption requirements for the VSC7430 device.

6.3.1 Current Consumption

The following tables show the operating current for the device. Typical current consumption values are over a full traffic load, nominal process and supply voltages, and at 25 °C case temperature. Maximum current consumption values are over full traffic load and worst-case process, temperature, and supply settings.

The following table shows current consumption values for the device.

Table 309 • **Operating Current**

Parameter	Symbol	Typical	Maximum (T_{J_MAX} = 110 °C)	Maximum (T_{J_MAX} = 125 °C)	Unit
V_{DD} operating current, 1.0 V	I_{DD}	1.3	2.8	3.3	A
V_{DD_A} operating current, 1.0 V	I_{DD_A}	0.4	0.4	0.4	A
V_{DD_HS} operating current, 1.0 V	I_{DD_HS}	0.01	0.01	0.01	A
V_{DD_VS} operating current, 1.0 V or 1.2 V	I_{DD_VS}	0.15	0.15	0.15	A
V_{DD_AL} operating current, 1.0 V	I_{DD_AL}	0.04	0.06	0.06	A
V_{DD_AH} operating current, 2.5 V	I_{DD_AH}	0.25	0.25	0.25	A
V_{DD_IODDR} operating current[1], 1.35 V or 1.5 V	I_{DD_IODDR}	0.08	0.18	0.18	A
V_{DD_IO} operating current[2], 2.5 V	I_{DD_IO}	0.05	0.065	0.065	A

1. DDR3 on-die termination is disabled.
2. Unloaded pins.

6.3.2 Power Consumption

The following table shows power consumption values when the device is at maximum configuration; all interfaces and functions enabled (2 × cuPHY + 2 × GbE (dual media), 2 × GbE, 2 × 2.5 GbE).

Table 310 • **Power Consumption**

Parameter	Typical	Maximum (T_{J_MAX} = 110 °C)	Maximum (T_{J_MAX} = 125 °C)	Unit
Power consumption, clock frequency = 156 MHz	2.8	4.7	5.0	W
Power consumption, clock frequency = 52 MHz	2.1	3.7	3.9	W

The following table shows typical power consumption reductions for reduced configurations compared to maximum configurations.

Table 311 • **Power Consumption, Reduced Configuration**

Parameter	Typical	Unit
Power consumption reduction for each cuPHY port disabled	0.3	W

6.3.3 Power Supply Sequencing

During power on and off, V_{DD_A}, V_{DD_HS}, and V_{DD_VS} must never be more than 300 mV above V_{DD}.

V_{DD_HS} and V_{DD_VS} must be powered, even if the associated interface is not used. These power supplies must not remain at ground or left floating.

A maximum delay of 100 ms from V_{DD_IODDR} to V_{DD} is recommended. There is no requirement from V_{DD} to V_{DD_IODDR}.

The V_{DD_IODDR} supply can remain at ground or left floating if not used. If V_{DD_IODDR} is grounded, DDR_Vref must also be grounded.

There are no sequencing requirements for V_{DD_IO}.

The nRESET and JTAG_nTRST inputs must be held low until all power supply voltages have reached their recommended operating condition values.

6.4 Operating Conditions

The following table shows the recommended operating conditions.

Table 312 • **Recommended Operating Conditions**

Parameter	Symbol	Minimum	Typical	Maximum	Unit
Power supply voltage for core supply	V_{DD}	0.95	1.0	1.05	V
Power supply voltage for analog circuits	V_{DD_A}	0.95	1.0	1.05	V
Power supply voltage for analog circuits[1], 1.0 V	V_{DD_HS}	0.95	1.0	1.05	V
Power supply voltage for 1G and 6G interfaces, 1.0 V[2]	V_{DD_VS}	0.95	1.0	1.05	V
Power supply voltage for 1G and 6G interfaces, 1.2 V	V_{DD_VS}	1.14	1.2	1.26	V
Power supply voltage for copper PHY analog circuits	V_{DD_AL}	0.95	1.0	1.05	V
Power supply voltage for copper PHY interface	V_{DD_AH}	2.375	2.5	2.625	V
Power supply voltage for DDR3 interface	V_{DD_IODDR}	1.425	1.5	1.575	V
Power supply voltage for DDR3L interface	V_{DD_IODDR}	1.28	1.35	1.45	V
Power supply voltage for GPIO and miscellaneous I/O	V_{DD_IO}	2.375	2.5	2.625	V

Table 312 • **Recommended Operating Conditions *(continued)***

Parameter	Symbol	Minimum	Typical	Maximum	Unit
Operating temperature[3]	T	–40		125	°C

1. V_{DD_HS} can be connected to V_{DD_A}.
2. V_{DD_VS} should be powered with the 1.2 V supply to meet 1000BASE-KX, XAUI, 10GBASE-CX4, 10GBASE-KX4, and PCIe specifications
3. Minimum specification is ambient temperature, and the maximum is junction temperature.

6.5 Stress Ratings

Warning Stresses listed in the following table may be applied to devices one at a time without causing permanent damage. Functionality at or exceeding the values listed is not implied. Exposure to these values for extended periods may affect device reliability.

Table 313 • **Stress Ratings**

Parameter	Symbol	Minimum	Maximum	Unit
Power supply voltage for core supply	V_{DD}	–0.3	1.10	V
Power supply voltage for analog circuits	V_{DD_A}	–0.3	1.10	V
Power supply voltage for analog circuits	V_{DD_HS}	–0.3	1.32	V
Power supply voltage for 1G and 6G interfaces	V_{DD_VS}	–0.3	1.32	V
Power supply voltage for copper PHY analog circuits	V_{DD_AL}	–0.3	1.10	V
Power supply voltage for copper PHY interface	V_{DD_AH}	–0.3	2.75	V
Power supply voltage for DDR I/O buffers	V_{DD_IODDR}	–0.3	1.98	V
Power supply voltage for GPIO and miscellaneous I/O	V_{DD_IO}	–0.3	2.75	V
Storage temperature	T_S	–55	125	°C
Electrostatic discharge voltage, charged device model	V_{ESD_CDM}	–500	500	V
Electrostatic discharge voltage, human body model, CLKOUTPLL2_N and CLKOUTPLL2_P pins	V_{ESD_HBM}	–1750	1750	V
Electrostatic discharge voltage, human body model, all pins except the CLKOUTPLL2_N and CLKOUTPLL2_P pins	V_{ESD_HBM}	See note[1]		V

1. This device has completed all required testing as specified in the JEDEC standard JESD22-A114, *Electrostatic Discharge (ESD) Sensitivity Testing Human Body Model (HBM)*, and complies with a Class 2 rating. The definition of Class 2 is any part that passes an ESD pulse of 2000 V, but fails an ESD pulse of 4000 V.

Warning This device can be damaged by electrostatic discharge (ESD) voltage. Microsemi recommends that all integrated circuits be handled with appropriate precautions. Failure to observe proper handling and installation procedures may adversely affect reliability of the device.

7 Pin Descriptions

The VSC7430 device has 324 pins, which are described in this section.

The pin information is also provided as an attached Microsoft Excel file, so that you can copy it electronically. In Adobe Reader, double-click the attachment icon.

7.1 Pin Diagram

The following illustration is a representation of the VSC7430 device, as seen from the top view looking through the device. For clarity, the device is shown in two halves, the top left and the top right.

Figure 188 • **Pin Diagram, Top Left**

	1	2	3	4	5	6	7	8	9
A	VSS_65	REFCLK_N	REFCLK_P	S5_RXN	S4_RXN	S3_RXN	S2_RXN	S1_RXN	S0_RXN
B	CLKOUTPLL_N	CLKOUTPLL_P	S6_TXN	S5_RXP	S4_RXP	S3_RXP	S2_RXP	S1_RXP	S0_RXP
C	S6_RXN	S6_RXP	S6_TXP	S5_TXN	S4_TXN	S3_TXN	S2_TXN	S1_TXN	S0_TXN
D	PCIE_TXN	PCIE_TXP	VDD_VS_1	S5_TXP	S4_TXP	S3_TXP	S2_TXP	S1_TXP	S0_TXP
E	PCIE_RXN	PCIE_RXP	VDD_VS_2	VDD_VS_3	VDD_VS_4	VDD_VS_5	VDD_VS_6	VDD_1	VDD_2
F	RESERVED_17	RESERVED_18	RESERVED_19	RESERVED_20	VDD_A_1	VDD_A_2	VSS_1	VSS_2	VSS_3
G	RESERVED_13	RESERVED_14	RESERVED_15	RESERVED_16	VDD_A_3	VSS_7	VSS_8	VSS_9	VSS_10
H	VSS_14	VSS_15	VDD_HS_1	VDD_A_4	VDD_11	VSS_16	VSS_17	VSS_18	VSS_19
J	RESERVED_11	RESERVED_12	VSS_23	VDD_A_5	VDD_13	VSS_24	VSS_25	VSS_26	VSS_27
K	VSS_31	VSS_32	VDD_HS_2	VDD_A_6	VDD_15	VSS_33	VSS_34	VSS_35	VSS_36
L	RESERVED_9	RESERVED_10	VSS_40	VDD_A_7	VDD_17	VSS_41	VSS_42	VSS_43	VSS_44
M	VSS_48	VSS_49	VDD_HS_3	VDD_A_8	VDD_19	VSS_50	VSS_51	VSS_52	VSS_53
N	RESERVED_7	RESERVED_8	VSS_57	VDD_A_9	VDD_22	VDD_23	VDD_24	VDD_25	VDD_26
P	VSS_58	VSS_59	VDD_HS_4	RESERVED_4	RESERVED_3	VDD_IO_1	VDD_IO_2	VDD_IO_3	VDD_IO_4
R	RESERVED_5	RESERVED_6	VSS_60	JTAG_nTRST	VSS_64	SI_DI	GPIO_0	GPIO_4	GPIO_8
T	VSS_61	VSS_62	VDD_HS_5	JTAG_TMS	JTAG_ICE_nEN	SI_DO	GPIO_1	GPIO_5	GPIO_9
U	REFCLK2_N	REFCLK2_P	VSS_63	JTAG_TDI	JTAG_TDO	SI_nCS0	GPIO_2	GPIO_6	GPIO_10
V	VSS_67	CLKOUTPLL2_N	CLKOUTPLL2_P	JTAG_TCK	nRESET	SI_CLK	GPIO_3	GPIO_7	GPIO_11

Figure 189 **• Pin Diagram, Top Right**

10	11	12	13	14	15	16	17	18	
P1_D0N	P1_D1N	P1_D2N	P1_D3N	P0_D0N	P0_D1N	P0_D2N	P0_D3N	VSS_66	A
P1_D0P	P1_D1P	P1_D2P	P1_D3P	P0_D0P	P0_D1P	P0_D2P	P0_D3P	RESERVED_0	B
VDD_AL_1	VDD_AL_2	VDD_AL_3	VDD_AL_4	VDD_AH_1	VDD_AH_2	VDD_AH_3	VDD_AH_4	RESERVED_1	C
SERDES_Rext_1	SERDES_Rext_0	REF_Rext	REF_FILT	THERMDC	THERMDA	DDR_DQ13	DDR_DQ10	DDR_DQ8	D
VDD_3	VDD_4	VDD_5	VDD_6	VDD_7	DDR_DQ15	DDR_UDM	DDR_UDQSn	DDR_UDQS	E
VSS_4	VSS_5	VSS_6	VDD_8	VDD_9	DDR_DQ14	DDR_DQ9	DDR_DQ11	DDR_DQ12	F
VSS_11	VSS_12	VSS_13	VDD_10	VDD_IODDR_1	DDR_nCS1	RESERVED_2	DDR_nWE	DDR_CKE	G
VSS_20	VSS_21	VSS_22	VDD_12	VDD_IODDR_2	DDR_BA2	DDR_BA0	DDR_BA1	DDR_A10	H
VSS_28	VSS_29	VSS_30	VDD_14	VDD_IODDR_3	DDR_A1	DDR_A3	DDR_A5	DDR_A7	J
VSS_37	VSS_38	VSS_39	VDD_16	VDD_IODDR_4	DDR_A9	DDR_A12	DDR_CK	DDR_CKn	K
VSS_45	VSS_46	VSS_47	VDD_18	VDD_IODDR_5	DDR_A11	DDR_A4	DDR_A6	DDR_A8	L
VSS_54	VSS_55	VSS_56	VDD_20	VDD_21	DDR_A13	DDR_nCS0	DDR_nCAS	DDR_A0	M
VDD_27	VDD_28	VDD_29	VDD_30	VDD_31	DDR_A2	DDR_A15	DDR_A14	DDR_nRAS	N
VDD_IO_5	VDD_IO_6	VDD_IO_7	VDD_IO_8	VDD_32	DDR_ODT	DDR_DQ7	DDR_DQ0	DDR_DQ2	P
GPIO_12	GPIO_16	GPIO_20	GPIO_24	GPIO_28	DDR_DQ5	DDR_DQ6	DDR_LDQS	DDR_LDQSn	R
GPIO_13	GPIO_17	GPIO_21	GPIO_25	GPIO_29	GPIO_32	DDR_LDM	DDR_DQ4	DDR_DQ1	T
GPIO_14	GPIO_18	GPIO_22	GPIO_26	GPIO_30	GPIO_33	GPIO_35	DDR_Rext	DDR_DQ3	U
GPIO_15	GPIO_19	GPIO_23	GPIO_27	GPIO_31	GPIO_34	GPIO_36	DDR_Vref	VSS_68	V

7.2 Pins by Function

This section contains the functional pin descriptions for the VSC7430 device.

The following table lists the definitions for the pin type symbols.

Table 314 **• Pin Type Symbol Definitions**

Symbol	Pin Type	Description
A	Analog input	Analog input for sensing variable voltage levels
ABIAS	Analog bias	Analog bias pin
DIFF	Differential	Differential signal pair
I	Input	Input signal
O	Output	Output signal
I/O	Bidirectional	Bidirectional input or output signal
OZ	3-state output	Output
LVDS	Input or output	Low voltage differential signal
LVCMOS	Input or output	Low voltage CMOS signal
3V		3.3 V-tolerant
ST	Schmitt-trigger	Input has Schmitt-trigger circuitry

Table 314 • **Pin Type Symbol Definitions *(continued)***

Symbol	Pin Type	Description
TD	Termination differential	Internal differential termination

7.2.1 DDR SDRAM Interface

The following table lists the pins associated with the DDR3/DDR3L SDRAM interface.

Table 315 • **DDR3/DDR3L SDRAM Pins**

Name	I/O	Type	Level	Description
DDR_A[15:0]	O		SSTL	SDRAM address outputs. Provide row and column addresses to the SDRAM.
DDR_BA[2:0]	O		SSTL	SDRAM bank address outputs. DDR_BA[2:0] define to which bank an ACTIVE, READ, WRITE, or PRECHARGE command is applied.
DDR_CK DDR_CKn	O	DIFF	SSTL	SDRAM differential clock. Differential clock to external SDRAM. DDR_CK is the true part of the differential signal. DDR_CKn is the complement part.
DDR_CKE	O		SSTL	SDRAM clock enable. 0: Disables clock in external SDRAM. 1: Enables clock in external SDRAM.
DDR_nCS[1:0]	O		SSTL	SDRAM chip selects. Active low.
DDR_DQ[15:0]	I/O		SSTL	SDRAM data bus.
DDR_LDM DDR_UDM	O		SSTL	SDRAM data mask outputs. DDR_LDM and DDR_UDM are mask signals for data written to the SDRAM. DDR_LDM corresponds to data on DDR_DQ[7:0], and DDR_UDM corresponds to data on DDR_DQ[15:8].
DDR_LDQS DDR_LDQSn DDR_UDQS DDR_UDQSn	I/O	DIFF, TD	SSTL	SDRAM differential data strobes. Bidirectional differential signal that follows data direction. Used by SDRAM to capture data on write operations. Edge-aligned with data from SDRAM during read operations and used by the device to capture data. DDR_LDQS corresponds to data on DDR_DQ[7:0], and DDR_UDQS corresponds to data on DDR_DQ[15:8].
DDR_nRAS DDR_nCAS DDR_nWE	O		SSTL	SDRAM command outputs. DDR_nRAS, DDR_nCAS, and DDR_nWE (along with DDR_nCS) define the command being entered.
DDR_ODT	O		SSTL	Control signal for the attached DDR3/DDR3L SDRAM devices on-die termination.
DDR_Rext		ABIAS	Analog	External DDR impedance calibration. Connect the pin through an external 240 Ω ±1% resistor to ground.
DDR_Vref		ABIAS	Analog	Input reference voltage. Provides the input switching reference voltage to the SSTL DDR signals.

7.2.2 General-Purpose Inputs and Outputs

The following table lists the general-purpose I/O (GPIO) pins. Leave GPIO pins unconnected when not in use.

Many of the GPIO pins serve multiple functions. For more information about the overlaid functions and how to configure them, GPIO Overlaid Functions, page 439.

Table 316 • **GPIO Pins**

Name	I/O	Level	Description
GPIO_[36:0]	I/O	CMOS	General-purpose inputs and outputs.

7.2.3 JTAG Interface

The following table lists the pins associated with the JTAG interface. The JTAG interface can be connected to the boundary scan TAP controller or the internal VCore-III TAP controller for software debug as described under the JTAG_ICE_nEN signal.

The JTAG signals are not 5 V tolerant.

Table 317 • **JTAG Interface Pins**

Name	I/O	Type	Level	Description
JTAG_ICE_nEN	I	ST, 3V	CMOS	0: Enables VCore-III debug interface over the JTAG interface. 1: Enables normal JTAG I/O over the JTAG interface. JTAG_ICE_nEN should be pulled either high or low using a resistor of 10 kΩ or less to enable the desired operating mode.
JTAG_nTRST	I	ST, 3V	CMOS	JTAG test reset, active low. For normal device operation, JTAG_nTRST should be pulled low.
JTAG_TCK	I	ST, 3V	CMOS	JTAG clock.
JTAG_TDI	I	ST, 3V	CMOS	JTAG test data in.
JTAG_TDO	O	OZ, 3V	CMOS	JTAG test data out.
JTAG_TMS	I	ST, 3V	CMOS	JTAG test mode select.

7.2.4 MII Management Interface

The following table lists the pins associated with the MII Management interface. The MIIM signals are GPIO alternate functions. For more information about the GPIO pin mapping, see GPIO Overlaid Functions, page 439.

Table 318 • **MII Management Interface Pins**

Name	I/O	Type	Level	Description
MDC1	I/O	3V	CMOS	Management data clock. MDC1 is sourced by the station management entity (the device) to the PHY as the timing reference for transfer of information on the MDIO1 signal. MDC1 is an aperiodic signal.
MDIO1	I/O	3V	CMOS	Management data input/output. MDIO1 is a bidirectional signal between a PHY and the device, used to transfer control and status information. Control information is driven by the device synchronously with respect to MDC1 and is sampled synchronously by the PHY. Status information is driven by the PHY synchronously with respect to MDC1 and is sampled synchronously by the device.

7.2.5 Miscellaneous

The following table lists the pins associated with a particular interface or facility on the device.

Table 319 • **Miscellaneous Pins**

Name	I/O	Type	Level	Description
nRESET	I	3V	CMOS	Global device reset, active low.
REF_FILT		ABIAS	Analog	Copper media reference filter pin. Connect a 1.0 µF external capacitor between pin and ground.
REF_Rext		ABIAS	Analog	Copper media reference external pin. Connect a 2.0 kΩ (1%) resistor between pin and ground.
RESERVED_0 RESERVED_1		Analog	Analog	Leave floating or connect to test point.
RESERVED_2	O		SSTL	Leave floating.
RESERVED_3 RESERVED_4		Analog	Analog	Tie to V_{SS}.
RESERVED_6:5 RESERVED_10:9 RESERVED_14:13 RESERVED_18_17	I		CML	Tie to V_{DD_A}.
RESERVED_8:7 RESERVED_12:11 RESERVED_16:15 RESERVED_20:19	O		CML	Leave floating.
SERDES_Rext_[1:0]		Analog	Analog	Analog bias calibration. Connect an external 620 Ω ±1% resistor between SERDES_Rext_1 and SERDES_Rext_0.
THERMDA		Analog	Analog	Thermal diode anode (p-junction).
THERMDC		Analog	Analog	Thermal diode cathode (n-junction). Connected on-die to V_{SS}.

7.2.6 PCI Express Interface

The following table lists the pins associated with PCI Express (PCIe).

Table 320 • **PCI Express Interface Pins**

Name	I/O	Type	Level	Description
PCIE_RXN PCIE_RXP	I	DIFF, TD	CML	Differential PCIe data inputs.
PCIE_TXN PCIE_TXP	O	DIFF	CML	Differential PCIe data outputs.

7.2.7 Power Supplies and Ground

The following table lists the power supply and ground pins.

Table 321 • **Power Supply and Ground Pins**

Name	Type	Description
VDD_[32:1]	Power	1.0 V power supply voltage for core
VDD_A_[9:1]	Power	1.0 V power supply voltage for analog circuits
VDD_AH_[4:1]	Power	2.5 V power supply for copper PHY interface
VDD_AL_[4:1]	Power	1.0 V power supply for copper PHY analog circuits
VDD_HS_[5:1]	Power	1.0 V power supply for analog circuits[1]
VDD_IO_[8:1]	Power	2.5 V power supply for GPIO and miscellaneous I/Os
VDD_IODDR_[5:1]	Power	1.35 V or 1.5 V power supply for DDR3/DDR3L interface
VDD_VS_[6:1]	Power	1.0 V or 1.2 V power supply voltage for SERDES1G and SERDES6G
VSS_[68:1]	Ground	Ground reference

1. V_{DD_HS} can be connected to V_{DD_A}.

7.2.8 SERDES1G

The following table lists the pins that use the SERDES1G differential macro. For information about the protocols and data rates supported by the SERDES1G macro, see Table 6, page 34.

Table 322 • **SERDES1G Pins**

Name	I/O	Type	Level	Description
S4_RXN S4_RXP	I	TD	LVDS	Differential 1G NPI data inputs.
S4_TXN S4_TXP	O		LVDS	Differential 1G NPI data outputs.
S[3:2]_RXN S[3:2]_RXP	I	TD	LVDS	Differential 1G data inputs.
S[3:2]_TXN S[3:2]_TXP	O		LVDS	Differential 1G data outputs.
S[1:0]_RXN S[1:0]_RXP	I	TD	LVDS	Differential 1G data inputs/dual media.
S[1:0]_TXN S[1:0]_TXP	O		LVDS	Differential 1G data outputs/dual media.

7.2.9 SERDES6G

The following table lists the pins that use the SERDES6G differential macro. For information about the protocols and data rates supported by the SERDES6G macro, see Table 6, page 34.

Table 323 • **SERDES6G Pins**

Name	I/O	Type	Level	Description
S[6:5]_RXN S[6:5]_RXP	I	TD	CML	Differential data inputs. 1G/2.5G

Table 323 • **SERDES6G Pins *(continued)***

Name	I/O	Type	Level	Description
S[6:5]_TXN S[6:5]_TXP	O		CML	Differential data outputs. 1G/2.5G

7.2.10 Serial CPU Interface

The serial CPU interface (SI) can be used as a serial slave, master, or boot interface.

As a slave interface, it allows external CPUs to access internal registers. As a master interface, it allows the device to access external devices using a programmable protocol. The serial CPU interface also allows the internal VCore-III CPU system to boot from an attached serial memory device when the VCore_CFG signals are set appropriately.

The following table lists the pins associated with the serial CPU interface (SI).

Table 324 • **Serial CPU Interface Pins**

Name	I/O	Type	Level	Description
SI_CLK	I/O	ST, 3V	LVCMOS	Slave mode: Input receiving serial interface clock from external master. Master mode: Output driven with clock to external device. Boot mode: Output driven with clock to external serial memory device.
SI_DI	I	ST, 3V	LVCMOS	Slave mode: Input receiving serial interface data from external master. Master mode: Input data from external device. Boot mode: Input boot data from external serial memory device.
SI_DO	O	OZ, 3V	LVCMOS	Slave mode: Output transmitting serial interface data to external master. Master mode: Output data to external device. Boot mode: Output boot control data to external serial memory device.
SI_nCS0	I/O	ST, 3V	LVCMOS	Slave mode: Input used to enable SI slave interface. 0 = Enabled 1 = Disabled Master mode: Output driven low while accessing external device. Boot mode: Output driven low while booting from EEPROM or serial flash to internal VCore-III CPU system. Released when booting is completed.

7.2.11 System Clock Interface

The following table lists the pins associated with the system clock interface.

Table 325 • **System Clock Interface Pins**

Name	I/O	Type	Level	Description
CLKOUTPLL_N CLKOUTPLL_P	O		LVDS	PLL clock output. Leave floating if not used.
CLKOUTPLL2_N CLKOUTPLL2_P	O		LVDS	PLL2 clock output. Leave floating if not used.

Table 325 • **System Clock Interface Pins *(continued)***

Name	I/O	Type	Level	Description
REFCLK_N	I	TD	LVDS	Reference clock inputs.
REFCLK_P			LVCMOS	The inputs can be either differential or single-ended.
REFCLK2_N REFCLK2_P				In differential mode (LVDS), REFCLK_P/REFCLK2_P is the true part of the differential signal, and REFCLK_N/REFCLK_N is the complement part of the differential signal. In single-ended mode (LVCMOS), REFCLK_P/REFCLK2_P is used as single-ended LVTTL input. REFCLK_N/REFCLK2_N should be left floating, and the PLL registers must be configured for single-ended operation. Required applied frequency depends on REFCLK_CONF[2:0] and REFCLK2_CONF[2:0] input state.

7.2.12 Twisted Pair Interface

The following table lists the pins associated with the twisted pair interface.

Table 326 • **Twisted Pair Interface Pins**

Name	I/O	Type	Description
P0_D0P P1_D0P	I/O	A_{DIFF}	Tx/Rx channel A positive signal. Positive differential signal connected to the positive primary side of the transformer. This pin signal forms the positive signal of the A data channel. In all three speeds, these pins generate the secondary side signal, normally connected to RJ-45 pin 1.
P0_D0N P1_D0N	I/O	A_{DIFF}	Tx/Rx channel A negative signal. Negative differential signal connected to the negative primary side of the transformer. This pin signal forms the negative signal of the A data channel. In all three speeds, these pins generate the secondary side signal, normally connected to RJ-45 pin 2.
P0_D1P P1_D1P	I/O	A_{DIFF}	Tx/Rx channel B positive signal. Positive differential signal connected to the positive primary side of the transformer. This pin signal forms the positive signal of the B data channel. In all three speeds, these pins generate the secondary side signal, normally connected to RJ-45 pin 3.
P0_D1N P1_D1N	I/O	A_{DIFF}	Tx/Rx channel B negative signal. Negative differential signal connected to the negative primary side of the transformer. This pin signal forms the negative signal of the B data channel. In all three speeds, these pins generate the secondary side signal, normally connected to RJ-45 pin 6.
P0_D2P P1_D2P	I/O	A_{DIFF}	Tx/Rx channel C positive signal. Positive differential signal connected to the positive primary side of the transformer. This pin signal forms the positive signal of the C data channel. In 1000-Mbps mode, these pins generate the secondary side signal, normally connected to RJ-45 pin 4 (pins not used in 10/100 Mbps modes).
P0_D2N P1_D2N	I/O	A_{DIFF}	Tx/Rx channel C negative signal. Negative differential signal connected to the negative primary side of the transformer. This pin signal forms the negative signal of the C data channel. In 1000-Mbps mode, these pins generate the secondary side signal, normally connected to RJ-45 pin 5 (pins not used in 10/100 Mbps modes).

Table 326 • **Twisted Pair Interface Pins *(continued)***

Name	I/O	Type	Description
P0_D3P	I/O	A_{DIFF}	Tx/Rx channel D positive signal.
P1_D3P			Positive differential signal connected to the positive primary side of the transformer. This pin signal forms the positive signal of the D data channel. In 1000-Mbps mode, these pins generate the secondary side signal, normally connected to RJ-45 pin 7 (pins not used in 10/100 Mbps modes).
P0_D3N	I/O	A_{DIFF}	Tx/Rx channel D negative signal.
P1_D3N			Negative differential signal connected to the negative primary side of the transformer. This pin signal forms the negative signal of the D data channel. In 1000-Mbps mode, these pins generate the secondary side signal, normally connected to RJ-45 pin 8 (pins not used in 10/100 Mbps modes).

7.3 234Pins by Number

This section provides a numeric list of the pins.

A1	VSS_65
A2	REFCLK_N
A3	REFCLK_P
A4	S5_RXN
A5	S4_RXN
A6	S3_RXN
A7	S2_RXN
A8	S1_RXN
A9	S0_RXN
A10	P1_D0N
A11	P1_D1N
A12	P1_D2N
A13	P1_D3N
A14	P0_D0N
A15	P0_D1N
A16	P0_D2N
A17	P0_D3N
A18	VSS_66
B1	CLKOUTPLL_N
B2	CLKOUTPLL_P
B3	S6_TXN
B4	S5_RXP
B5	S4_RXP
B6	S3_RXP
B7	S2_RXP
B8	S1_RXP
B9	S0_RXP
B10	P1_D0P
B11	P1_D1P
B12	P1_D2P
B13	P1_D3P
B14	P0_D0P
B15	P0_D1P
B16	P0_D2P
B17	P0_D3P
B18	RESERVED_0
C1	S6_RXN
C2	S6_RXP
C3	S6_TXP
C4	S5_TXN
C5	S4_TXN
C6	S3_TXN
C7	S2_TXN
C8	S1_TXN
C9	S0_TXN
C10	VDD_AL_1
C11	VDD_AL_2
C12	VDD_AL_3
C13	VDD_AL_4
C14	VDD_AH_1
C15	VDD_AH_2
C16	VDD_AH_3
C17	VDD_AH_4
C18	RESERVED_1
D1	PCIE_TXN
D2	PCIE_TXP
D3	VDD_VS_1
D4	S5_TXP
D5	S4_TXP
D6	S3_TXP
D7	S2_TXP
D8	S1_TXP
D9	S0_TXP
D10	SERDES_Rext_1
D11	SERDES_Rext_0
D12	REF_Rext
D13	REF_FILT
D14	THERMDC
D15	THERMDA
D16	DDR_DQ13
D17	DDR_DQ10
D18	DDR_DQ8
E1	PCIE_RXN
E2	PCIE_RXP
E3	VDD_VS_2
E4	VDD_VS_3
E5	VDD_VS_4
E6	VDD_VS_5
E7	VDD_VS_6
E8	VDD_1
E9	VDD_2
E10	VDD_3
E11	VDD_4
E12	VDD_5
E13	VDD_6
E14	VDD_7
E15	DDR_DQ15
E16	DDR_UDM
E17	DDR_UDQSn
E18	DDR_UDQS
F1	RESERVED_17
F2	RESERVED_18
F3	RESERVED_19
F4	RESERVED_20
F5	VDD_A_1
F6	VDD_A_2
F7	VSS_1
F8	VSS_2
F9	VSS_3
F10	VSS_4
F11	VSS_5
F12	VSS_6
F13	VDD_8
F14	VDD_9
F15	DDR_DQ14
F16	DDR_DQ9
F17	DDR_DQ11
F18	DDR_DQ12
G1	RESERVED_13
G2	RESERVED_14
G3	RESERVED_15
G4	RESERVED_16
G5	VDD_A_3
G6	VSS_7

Pins by number *(continued)*

G7	VSS_8
G8	VSS_9
G9	VSS_10
G10	VSS_11
G11	VSS_12
G12	VSS_13
G13	VDD_10
G14	VDD_IODDR_1
G15	DDR_nCS1
G16	RESERVED_2
G17	DDR_nWE
G18	DDR_CKE
H1	VSS_14
H2	VSS_15
H3	VDD_HS_1
H4	VDD_A_4
H5	VDD_11
H6	VSS_16
H7	VSS_17
H8	VSS_18
H9	VSS_19
H10	VSS_20
H11	VSS_21
H12	VSS_22
H13	VDD_12
H14	VDD_IODDR_2
H15	DDR_BA2
H16	DDR_BA0
H17	DDR_BA1
H18	DDR_A10
J1	RESERVED_11
J2	RESERVED_12
J3	VSS_23
J4	VDD_A_5
J5	VDD_13
J6	VSS_24
J7	VSS_25
J8	VSS_26
J9	VSS_27
J10	VSS_28
J11	VSS_29
J12	VSS_30
J13	VDD_14
J14	VDD_IODDR_3
J15	DDR_A1
J16	DDR_A3
J17	DDR_A5
J18	DDR_A7
K1	VSS_31
K2	VSS_32
K3	VDD_HS_2
K4	VDD_A_6
K5	VDD_15
K6	VSS_33
K7	VSS_34
K8	VSS_35
K9	VSS_36
K10	VSS_37
K11	VSS_38
K12	VSS_39
K13	VDD_16
K14	VDD_IODDR_4
K15	DDR_A9
K16	DDR_A12
K17	DDR_CK
K18	DDR_CKn
L1	RESERVED_9
L2	RESERVED_10
L3	VSS_40
L4	VDD_A_7
L5	VDD_17
L6	VSS_41
L7	VSS_42
L8	VSS_43
L9	VSS_44
L10	VSS_45
L11	VSS_46
L12	VSS_47
L13	VDD_18
L14	VDD_IODDR_5
L15	DDR_A11
L16	DDR_A4
L17	DDR_A6
L18	DDR_A8
M1	VSS_48
M2	VSS_49
M3	VDD_HS_3
M4	VDD_A_8
M5	VDD_19
M6	VSS_50
M7	VSS_51
M8	VSS_52
M9	VSS_53
M10	VSS_54
M11	VSS_55
M12	VSS_56
M13	VDD_20
M14	VDD_21
M15	DDR_A13
M16	DDR_nCS0
M17	DDR_nCAS
M18	DDR_A0
N1	RESERVED_7
N2	RESERVED_8
N3	VSS_57
N4	VDD_A_9
N5	VDD_22
N6	VDD_23
N7	VDD_24
N8	VDD_25
N9	VDD_26
N10	VDD_27
N11	VDD_28
N12	VDD_29
N13	VDD_30
N14	VDD_31
N15	DDR_A2
N16	DDR_A15
N17	DDR_A14
N18	DDR_nRAS
P1	VSS_58
P2	VSS_59
P3	VDD_HS_4

Pins by number *(continued)*

P4	RESERVED_4
P5	RESERVED_3
P6	VDD_IO_1
P7	VDD_IO_2
P8	VDD_IO_3
P9	VDD_IO_4
P10	VDD_IO_5
P11	VDD_IO_6
P12	VDD_IO_7
P13	VDD_IO_8
P14	VDD_32
P15	DDR_ODT
P16	DDR_DQ7
P17	DDR_DQ0
P18	DDR_DQ2
R1	RESERVED_5
R2	RESERVED_6
R3	VSS_60
R4	JTAG_nTRST
R5	VSS_64
R6	SI_DI
R7	GPIO_0
R8	GPIO_4
R9	GPIO_8
R10	GPIO_12
R11	GPIO_16
R12	GPIO_20
R13	GPIO_24
R14	GPIO_28
R15	DDR_DQ5
R16	DDR_DQ6
R17	DDR_LDQS
R18	DDR_LDQSn
T1	VSS_61
T2	VSS_62
T3	VDD_HS_5
T4	JTAG_TMS
T5	JTAG_ICE_nEN
T6	SI_DO
T7	GPIO_1
T8	GPIO_5
T9	GPIO_9
T10	GPIO_13
T11	GPIO_17
T12	GPIO_21
T13	GPIO_25
T14	GPIO_29
T15	GPIO_32
T16	DDR_LDM
T17	DDR_DQ4
T18	DDR_DQ1
U1	REFCLK2_N
U2	REFCLK2_P
U3	VSS_63
U4	JTAG_TDI
U5	JTAG_TDO
U6	SI_nCS0
U7	GPIO_2
U8	GPIO_6
U9	GPIO_10
U10	GPIO_14
U11	GPIO_18
U12	GPIO_22
U13	GPIO_26
U14	GPIO_30
U15	GPIO_33
U16	GPIO_35
U17	DDR_Rext
U18	DDR_DQ3
V1	VSS_67
V2	CLKOUTPLL2_N
V3	CLKOUTPLL2_P
V4	JTAG_TCK
V5	nRESET
V6	SI_CLK
V7	GPIO_3
V8	GPIO_7
V9	GPIO_11
V10	GPIO_15
V11	GPIO_19
V12	GPIO_23
V13	GPIO_27
V14	GPIO_31
V15	GPIO_34
V16	GPIO_36
V17	DDR_Vref
V18	VSS_68

7.4 Pins by Name

This section provides an alphabetic list of the pins.

CLKOUTPLL_N	B1
CLKOUTPLL_P	B2
CLKOUTPLL2_N	V2
CLKOUTPLL2_P	V3
DDR_A0	M18
DDR_A1	J15
DDR_A2	N15
DDR_A3	J16
DDR_A4	L16
DDR_A5	J17
DDR_A6	L17
DDR_A7	J18
DDR_A8	L18
DDR_A9	K15
DDR_A10	H18
DDR_A11	L15
DDR_A12	K16
DDR_A13	M15
DDR_A14	N17
DDR_A15	N16
DDR_BA0	H16
DDR_BA1	H17
DDR_BA2	H15
DDR_CK	K17
DDR_CKE	G18
DDR_CKn	K18
DDR_DQ0	P17
DDR_DQ1	T18
DDR_DQ2	P18
DDR_DQ3	U18
DDR_DQ4	T17
DDR_DQ5	R15
DDR_DQ6	R16
DDR_DQ7	P16
DDR_DQ8	D18
DDR_DQ9	F16
DDR_DQ10	D17
DDR_DQ11	F17
DDR_DQ12	F18
DDR_DQ13	D16
DDR_DQ14	F15
DDR_DQ15	E15
DDR_LDM	T16
DDR_LDQS	R17
DDR_LDQSn	R18
DDR_nCAS	M17
DDR_nCS0	M16
DDR_nCS1	G15
DDR_nRAS	N18
DDR_nWE	G17
DDR_ODT	P15
DDR_Rext	U17
DDR_UDM	E16
DDR_UDQS	E18
DDR_UDQSn	E17
DDR_Vref	V17
GPIO_0	R7
GPIO_1	T7
GPIO_2	U7
GPIO_3	V7
GPIO_4	R8
GPIO_5	T8
GPIO_6	U8
GPIO_7	V8
GPIO_8	R9
GPIO_9	T9
GPIO_10	U9
GPIO_11	V9
GPIO_12	R10
GPIO_13	T10
GPIO_14	U10
GPIO_15	V10
GPIO_16	R11
GPIO_17	T11
GPIO_18	U11
GPIO_19	V11
GPIO_20	R12
GPIO_21	T12
GPIO_22	U12
GPIO_23	V12
GPIO_24	R13
GPIO_25	T13
GPIO_26	U13
GPIO_27	V13
GPIO_28	R14
GPIO_29	T14
GPIO_30	U14
GPIO_31	V14
GPIO_32	T15
GPIO_33	U15
GPIO_34	V15
GPIO_35	U16
GPIO_36	V16
JTAG_ICE_nEN	T5
JTAG_nTRST	R4
JTAG_TCK	V4
JTAG_TDI	U4
JTAG_TDO	U5
JTAG_TMS	T4
nRESET	V5
P0_D0N	A14
P0_D0P	B14
P0_D1N	A15
P0_D1P	B15
P0_D2N	A16
P0_D2P	B16
P0_D3N	A17
P0_D3P	B17
P1_D0N	A10
P1_D0P	B10
P1_D1N	A11
P1_D1P	B11
P1_D2N	A12
P1_D2P	B12

Pins by name *(continued)*

P1_D3N	A13
P1_D3P	B13
PCIE_RXN	E1
PCIE_RXP	E2
PCIE_TXN	D1
PCIE_TXP	D2
REF_FILT	D13
REF_Rext	D12
REFCLK_N	A2
REFCLK_P	A3
REFCLK2_N	U1
REFCLK2_P	U2
RESERVED_0	B18
RESERVED_1	C18
RESERVED_2	G16
RESERVED_3	P5
RESERVED_4	P4
RESERVED_5	R1
RESERVED_6	R2
RESERVED_7	N1
RESERVED_8	N2
RESERVED_9	L1
RESERVED_10	L2
RESERVED_11	J1
RESERVED_12	J2
RESERVED_13	G1
RESERVED_14	G2
RESERVED_15	G3
RESERVED_16	G4
RESERVED_17	F1
RESERVED_18	F2
RESERVED_19	F3
RESERVED_20	F4
S0_RXN	A9
S0_RXP	B9
S0_TXN	C9
S0_TXP	D9
S1_RXN	A8
S1_RXP	B8
S1_TXN	C8
S1_TXP	D8
S2_RXN	A7
S2_RXP	B7
S2_TXN	C7
S2_TXP	D7
S3_RXN	A6
S3_RXP	B6
S3_TXN	C6
S3_TXP	D6
S4_RXN	A5
S4_RXP	B5
S4_TXN	C5
S4_TXP	D5
S5_RXN	A4
S5_RXP	B4
S5_TXN	C4
S5_TXP	D4
S6_RXN	C1
S6_RXP	C2
S6_TXN	B3
S6_TXP	C3
SERDES_Rext_0	D11
SERDES_Rext_1	D10
SI_CLK	V6
SI_DI	R6
SI_DO	T6
SI_nCS0	U6
THERMDA	D15
THERMDC	D14
VDD_1	E8
VDD_2	E9
VDD_3	E10
VDD_4	E11
VDD_5	E12
VDD_6	E13
VDD_7	E14
VDD_8	F13
VDD_9	F14
VDD_10	G13
VDD_11	H5
VDD_12	H13
VDD_13	J5
VDD_14	J13
VDD_15	K5
VDD_16	K13
VDD_17	L5
VDD_18	L13
VDD_19	M5
VDD_20	M13
VDD_21	M14
VDD_22	N5
VDD_23	N6
VDD_24	N7
VDD_25	N8
VDD_26	N9
VDD_27	N10
VDD_28	N11
VDD_29	N12
VDD_30	N13
VDD_31	N14
VDD_32	P14
VDD_A_1	F5
VDD_A_2	F6
VDD_A_3	G5
VDD_A_4	H4
VDD_A_5	J4
VDD_A_6	K4
VDD_A_7	L4
VDD_A_8	M4
VDD_A_9	N4
VDD_AH_1	C14
VDD_AH_2	C15
VDD_AH_3	C16
VDD_AH_4	C17
VDD_AL_1	C10
VDD_AL_2	C11
VDD_AL_3	C12
VDD_AL_4	C13
VDD_HS_1	H3
VDD_HS_2	K3
VDD_HS_3	M3
VDD_HS_4	P3
VDD_HS_5	T3

Pins by name *(continued)*

VDD_IO_1	P6
VDD_IO_2	P7
VDD_IO_3	P8
VDD_IO_4	P9
VDD_IO_5	P10
VDD_IO_6	P11
VDD_IO_7	P12
VDD_IO_8	P13
VDD_IODDR_1	G14
VDD_IODDR_2	H14
VDD_IODDR_3	J14
VDD_IODDR_4	K14
VDD_IODDR_5	L14
VDD_VS_1	D3
VDD_VS_2	E3
VDD_VS_3	E4
VDD_VS_4	E5
VDD_VS_5	E6
VDD_VS_6	E7
VSS_1	F7
VSS_2	F8
VSS_3	F9
VSS_4	F10
VSS_5	F11
VSS_6	F12
VSS_7	G6
VSS_8	G7
VSS_9	G8
VSS_10	G9
VSS_11	G10
VSS_12	G11
VSS_13	G12
VSS_14	H1
VSS_15	H2
VSS_16	H6
VSS_17	H7
VSS_18	H8
VSS_19	H9
VSS_20	H10
VSS_21	H11
VSS_22	H12
VSS_23	J3
VSS_24	J6
VSS_25	J7
VSS_26	J8
VSS_27	J9
VSS_28	J10
VSS_29	J11
VSS_30	J12
VSS_31	K1
VSS_32	K2
VSS_33	K6
VSS_34	K7
VSS_35	K8
VSS_36	K9
VSS_37	K10
VSS_38	K11
VSS_39	K12
VSS_40	L3
VSS_41	L6
VSS_42	L7
VSS_43	L8
VSS_44	L9
VSS_45	L10
VSS_46	L11
VSS_47	L12
VSS_48	M1
VSS_49	M2
VSS_50	M6
VSS_51	M7
VSS_52	M8
VSS_53	M9
VSS_54	M10
VSS_55	M11
VSS_56	M12
VSS_57	N3
VSS_58	P1
VSS_59	P2
VSS_60	R3
VSS_61	T1
VSS_62	T2
VSS_63	U3
VSS_64	R5
VSS_65	A1
VSS_66	A18
VSS_67	V1
VSS_68	V18

8 Package Information

The VSC7430XMT package is a lead-free (Pb-free), 324-pin, plastic ball grid array (BGA) with a 19 mm × 19 mm body size, 1 mm pin pitch, and 1.8 mm maximum height.

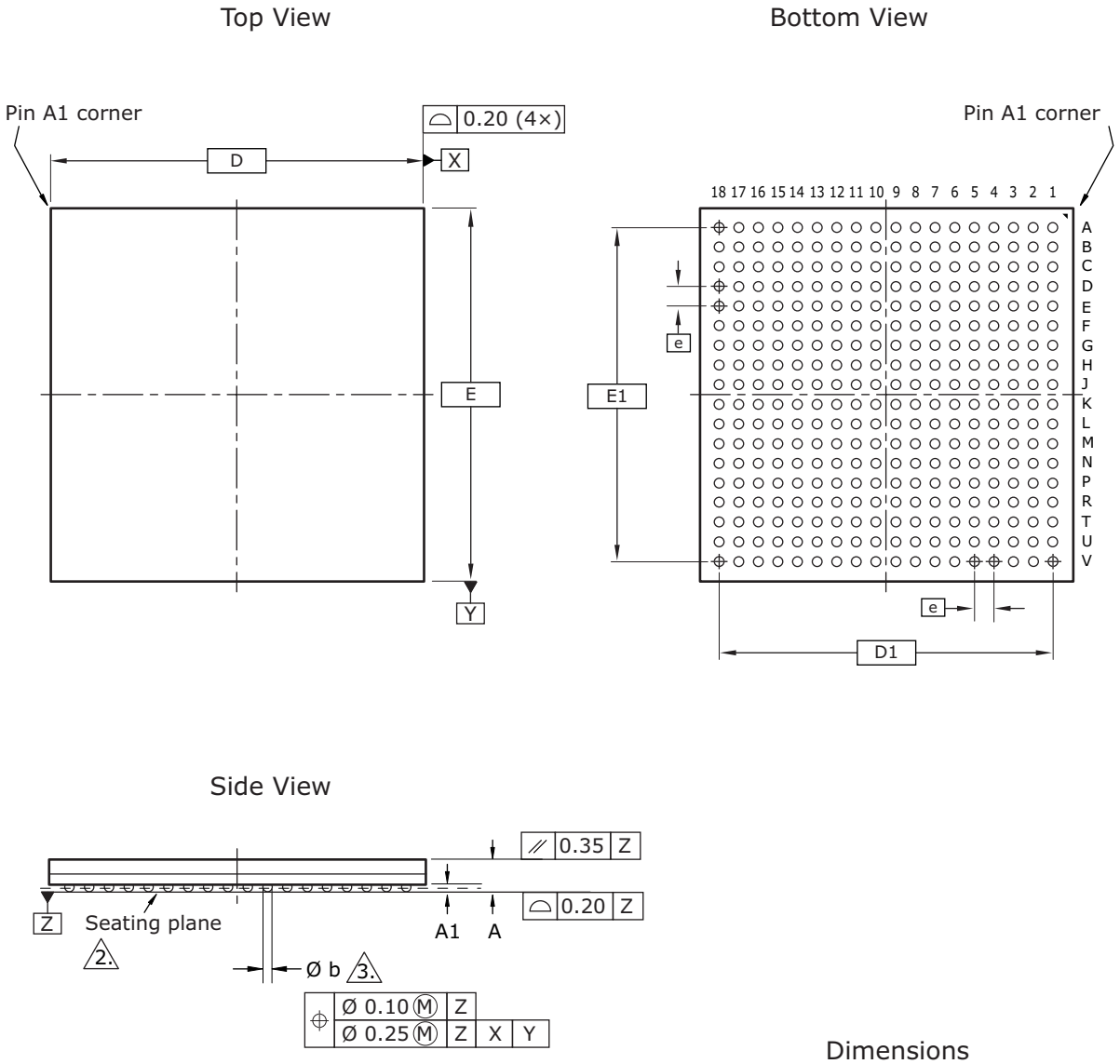
Lead-free products from Microsemi comply with the temperatures and profiles defined in the joint IPC and JEDEC standard IPC/JEDEC J-STD-020. For more information, see the IPC and JEDEC standard.

This section provides the package drawing, thermal specifications, and moisture sensitivity rating for the VSC7430 device.

8.1 Package Drawing

The following illustration shows the package drawing for the VSC7430 device. The drawing contains the top view, bottom view, side view, detail views, dimensions, tolerances, and notes.

***Figure 190* • Package Drawing**



Dimensions

Reference	Minimum	Nominal	Maximum
A	1.54	1.67	1.80
A1	0.31		0.43
D		19.00 BSC	
E		19.00 BSC	
D1		17.00 BSC	
E1		17.00 BSC	
e		1.00 BSC	
b	0.44		0.64

Notes

1. All dimensions and tolerances are in millimeters (mm).
2. Primary datum Z and seating plane are defined by the spherical crowns of the solder balls.
3. Dimension is measured at the maximum solder ball diameter, parallel to primary datum Z.
4. Radial true position is represented by typical values.

8.2 Thermal Specifications

Thermal specifications for this device are based on the JEDEC JESD51 family of documents. These documents are available on the JEDEC Web site at www.jedec.org. The thermal specifications are modeled using a four-layer test board with two signal layers, a power plane, and a ground plane (2s2p

PCB). For more information about the thermal measurement method used for this device, see the JESD51-1 standard.

Table 327 • **Thermal Resistances**

Symbol	°C/W	Parameter
θ_{JCtop}	2.89	Die junction to package case top
θ_{JB}	8.65	Die junction to printed circuit board
θ_{JA}	15.94	Die junction to ambient
θ_{JMA} at 1 m/s	13.3	Die junction to moving air measured at an air speed of 1 m/s
θ_{JMA} at 2 m/s	12.36	Die junction to moving air measured at an air speed of 2 m/s

To achieve results similar to the modeled thermal measurements, the guidelines for board design described in the JESD51 family of publications must be applied. For information about applications using BGA packages, see the following:

- JESD51-2A, *Integrated Circuits Thermal Test Method Environmental Conditions, Natural Convection (Still Air)*
- JESD51-6, *Integrated Circuit Thermal Test Method Environmental Conditions, Forced Convection (Moving Air)*
- JESD51-8, *Integrated Circuit Thermal Test Method Environmental Conditions, Junction-to-Board*
- JESD51-9, *Test Boards for Area Array Surface Mount Package Thermal Measurements*

8.3 Moisture Sensitivity

This device is rated moisture sensitivity level 4 as specified in the joint IPC and JEDEC standard IPC/JEDEC J-STD-020. For more information, see the IPC and JEDEC standard.

9 Design Guidelines

This section provides information about design guidelines for the VSC7430 device.

9.1 Power Supplies

The following guidelines apply to designing power supplies for use with the VSC7430 device.

- Make at least one unbroken ground plane (GND).
- Use the power and ground plane combination as an effective power supply bypass capacitor. The capacitance is proportional to the area of the two planes and inversely proportional to the separation between the planes. Typical values with a 0.25 mm (0.01 inch) separation are 100 pF/in2. This capacitance is more effective than a capacitor of equivalent value, because the planes have no inductance or Equivalent Series Resistance (ESR).
- Do not cut up the power or ground planes in an effort to steer current paths. This usually produces more noise, not less. Furthermore, place vias and clearances in a configuration that maintains the integrity of the plane. Groups of vias spaced close together often overlap clearances. This can form a large slot in the plane. As a result, return currents are forced around the slot, which increases the loop area and EMI emissions. Signals should never be placed on a ground plane, because the resulting slot forces return currents around the slot.
- Vias connecting power planes to the supply and ground balls must be at least 0.25 mm (0.010 inch) in diameter, preferably with no thermal relief and plated closed with copper or solder. Use separate (or even multiple) vias for each supply and ground ball.

9.2 Power Supply Decoupling

Each power supply voltage should have both bulk and high-frequency decoupling capacitors. Recommended capacitors are as follows:

- For bulk decoupling, use 10 µF high capacity and low ESR capacitors or equivalent, distributed across the board.
- For high-frequency decoupling, use 0.1 µF high frequency (for example, X7R) ceramic capacitors placed on the side of the PCB closest to the plane being decoupled, and as close as possible to the power ball. A larger value in the same housing unit produces even better results.
- Use surface-mounted components for lower lead inductance and pad capacitance. Smaller form factor components are best (that is, 0402 is better than 0603).

9.2.1 Reference Clock

The device reference clock can be a 25 MHz, 125 MHz, 156.25 MHz, or 250 MHz clock signal. It can be either a differential reference clock or a single-ended clock. For more information, see Reference Clock, page 458.

9.2.2 Single-Ended REFCLK Input

An external resistor network is required to use a single-ended reference clock. The network limits the amplitude and adjusts the center of the swing. Depending on the clock driver output impedance, the resistor values may need to be adjusted so that the REFCLK_P signal meets requirements. For information about these requirements, see Table 277, page 458.

The following illustrations show configurations for a single-ended reference clock.

Figure 191 • **2.5 V CMOS Single-Ended REFCLK Input Resistor Network**

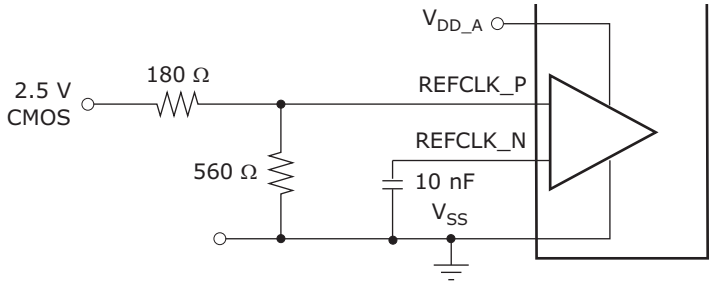
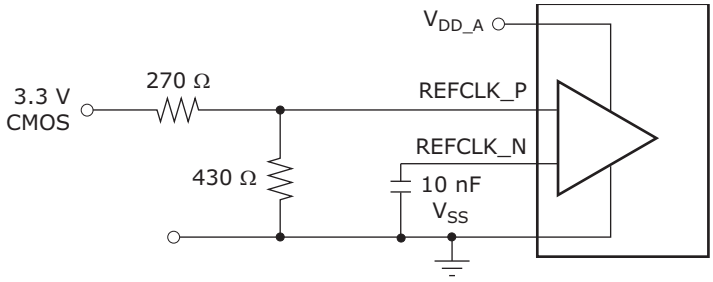


Figure 192 • **3.3 V CMOS Single-Ended REFCLK Input Resistor Network**



9.3 Interfaces

This section provides general recommendations for all interfaces and information related to the specific interfaces on the device.

9.3.1 General Recommendations

High-speed signals require excellent frequency and phase response up to the third harmonic. The best designs provide excellent frequency and phase response up to the seventh harmonic. The following recommendations can improve signal quality and minimize transmission distances:

- Keep traces as short as possible. Initial component placement should be considered very carefully.
- The impedance of the traces must match the impedance of the termination resistors, connectors, and cable. This reduces reflections due to impedance mismatches.
- Differential impedance must be maintained in a 100 Ω differential application. Routing two 50 Ω traces is not adequate. The two traces must be separated by enough distance to maintain differential impedance. When routing differential pairs, keep the two trace lengths identical. Differences in trace lengths translate directly into signal skew. Note that the differential impedance may be affected when separations occur.
- Keep differential pair traces on the same layer of the PCB to minimize impedance discontinuities. In other words, avoid using vias.
- Do not group all the passive components together. The pads of the components add capacitance to the traces. At the frequencies encountered, this can result in unwanted reductions in impedance. Use surface-mounted 0603 (or smaller) components to reduce this effect.
- Eliminate or reduce stub lengths.
- Reduce, if not eliminate, vias to minimize impedance discontinuities. Remember that vias and their clearance holes in the power/ground planes can cause impedance discontinuities in nearby signals. Keep vias away from other traces.
- Keep signal traces away from other signals that might capacitively couple noise into the signals. A good rule of thumb is to keep the traces apart by ten times the width of the trace.
- Do not route digital signals from other circuits across the area of the high-speed transmitter and receiver signals.
- Using grounded guard traces is typically not effective for improving signal quality. A ground plane is more effective. However, a common use of guard traces is to route them during the layout, but remove them prior to design completion. This has the benefit of enforcing keep-out areas around sensitive high-speed signals so that vias and other traces are not accidentally placed incorrectly.

- When signals in a differential pair are mismatched, the result is a common-mode current. In a well-designed system, common-mode currents should make up less than one percent of the total differential currents. Mode currents represent a primary source of EMI emissions. To reduce common-mode currents, route differential traces so that their lengths are the same. For example, a 5-mm (0.2-inch) length mismatch between differential signals having the rise and fall times of 200 ps results in the common-mode current being up to 18% of the differential current.

Note: Care must be taken when choosing proper components (such as the termination resistors) in the designing of the layout of a printed circuit board, because of the high application frequency. The use of surface-mount components is highly recommended to minimize parasitic inductance and lead length of the termination resistor.

Matching the impedance of the PCB traces, connectors, and balanced interconnect media is also highly recommended. Impedance variations along the entire interconnect path must be minimized, because they degrade the signal path and may cause reflections of the signal.

9.3.2 SerDes Interfaces (SGMII, 2.5G)

The SGMII interface consists of a Tx and Rx differential pair operating at 1250 Mbps. The 2.5G interface consists of a Tx and Rx differential pair operating at 3125 Mbps.

The SerDes signals can be routed on any PCB trace layer with the following constraints:

- Tx output signals in a pair should have matched electrical lengths.
- Rx input signals in a pair should have matched electrical lengths.
- SerDes Tx and Rx pairs must be routed as 100 Ω differential traces with ground plane as reference.
- Keep differential pair traces on the same layer of the PCB to minimize impedance discontinuities. In other words, avoid the use of vias wherever possible.
- AC-coupling of Tx and Rx may be needed, depending on the attached PHY or module. External AC-coupling is recommended for use with most PHYs. SFP modules have internal AC-coupling, so they do not require additional AC-coupling capacitors. If AC-coupled, the VSC7430 SerDes inputs are self-biased. It is recommended to use small form factor capacitors, 0402 or smaller, to reduce the impedance mismatch caused by the capacitor pads.
- To reduce the crosstalk between pairs or other PCB lines, it is recommended that the spacing on each side of the pair be larger than four times the track width. The characteristic impedance of the pairs must predominantly be determined by the distance to the reference plane, and not the distance to neighboring traces.

9.3.3 Serial Interface

If the serial CPU interface is not used, all input signals can be left floating.

The SI bus consists of the SI_CLK clock signal, the SI_DO and SI_DI data signals, and the SI_nCS0 device select signal.

When routing the SI_CLK signal, be sure to create clean edges. If the SI bus is connected to more than one slave device, route it in a daisy-chain configuration with no stubs. Terminate the SI_CLK signal properly to avoid reflections and double clocking.

If it is not possible (or desirable) to route the bus in a daisy-chain configuration, the SI_CLK signal should be buffered and routed in a star topology from the buffers. Each buffered clock should be terminated at its source.

9.3.4 PCI Express Interface

The VSC7430 device does not accept spread spectrum modulated PCIe input. Although the device only supports PCI Express Base Specification Revision 1.1, the PCIe transmitter and receiver support the PCI Express Base Specification Revision 2.0 Electrical sub-block specifications under the following conditions:

- Only 2.5 gigatransfers per second (GT/s) is supported.
- Full swing output signaling is only supported when V_{DD_VS} = 1.2 V.
- Low swing signaling is supported for V_{DD_VS} = 1.2 V and V_{DD_VS} = 1.0 V.

During PCIe startup (VCORE_CFG = 1001), only low swing signaling with no de-emphasis is supported, regardless of the V_{DD_VS} voltage. After startup, configure the PCIe interface as desired.

9.3.5 Two-Wire Serial Interface

The two-wire serial interface is capable of suppressing small amplitude glitches less than 5 ns in duration, which is less than the 50 ns duration often quoted for similar interfaces. Because the two-wire serial implementation uses Schmitt-triggered inputs, the VSC7430 device has a greater tolerance to low amplitude noise. For glitch-free operation, select the proper pull-up resistor value to ensure that the transition time through the input switching region is less than 5 ns given the line's total capacitive load. For capacitive loads up to 40 pF, a pull-up resistor of 510 Ω or less ensures glitch-free operation for noise levels up to 700 mV peak-to-peak.

9.3.6 DDR3 SDRAM Interface

The DDR3 SDRAM interface is designed to interface with 8-bit or 16-bit DDR SDRAM devices. The maximum amount of physical memory that can be addressed is one gigabyte. Possible combinations of memory modules are:

- One 8-bit device, connect to CS0 byte lane 0.
- Two 8-bit devices, connect to CS0 and both byte lanes.
- Four 8-bit devices, connect to both chip-selects and both byte lanes.
- One 16-bit device, connect to CS0 and both byte lanes.
- Two 16-bit devices, connect to both chip-selects and both byte lanes.

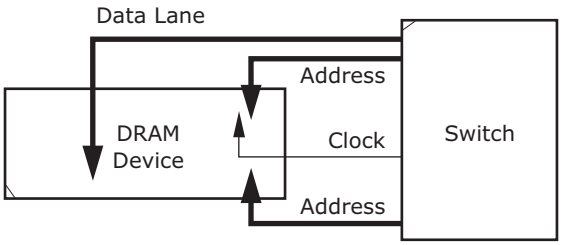
When using a single 8-bit device, the memory controller must be configured for 8-bit mode. All other configurations use 16-bit mode.

All signals on this interface must be connected one-to-one with the corresponding signals on the DDR SDRAM device. When using only one 8-bit device, the DDR_UDQS, DDR_UDQSn, DDR_UDM, and DDR_DQ[15:8] signals must be left unconnected. When using DDR2 SDRAM devices that has only four banks, the DDR_BA[2] signal must be left unconnected.

When using four 8-bit devices or two 16-bit devices, the CS1 memory modules must be placed in-between the device and the CS0 memory modules so that the DDR-DQS, DDR-DM, and DDR_DQ signals pass the CS1 devices first before reaching the CS0 devices.

The placement of the VSC7430 interface signals is optimized for point-to-point routing directly to a single DDR3 16-bit SDRAM.

***Figure 193* • 16-Bit DDR3 SDRAM Point-to-Point Routing**



Because reflections are absorbed by the devices, keep the physical distance of all the SDRAM interface signals as low as possible. Omit external discrete termination on the address, command, control and clock lines.

When routing the DDR interface, attention must be paid to the skew, primary concern is skew within the byte lane between the differential strobe and the single-ended signals. Skew recommendations for the DDR interface are listed in the following table.

***Table 328* • Recommended Skew Budget**

Description	Signal	Maximum Skew
Skew within byte lane 0	DDR_LDQS/DDR_LDQSn	50 ps

Table 328 • **Recommended Skew Budget**

Description	Signal	Maximum Skew
Skew within byte lane 1	DDR_UDQS/DDR_UDQSn	50 ps
Skew within address, command, and control bus	DDR_CK/DDR_CKn DDR_nRAS DDR_CKE DDR_ODT[1:0] DDR_nCAS DDR_nWE DDR_BA[2:0] DDR_A[15:0]	100 ps
Skew between control bus clock and byte lane 0 clock	DDR_CK/DDR_CKn DDR_LDQS/DDR_LDQSn	1250 ps
Skew between control bus clock and byte lane 1 clock	DDR_CK/DDR_CKn DDR_UDQS/DDR_UDQSn	1250 ps
Control bus differential clock intra-pair skew	DDR_CK/DDR_CKn	5 ps

Power supply recommendations:

- Use a shared voltage reference between the VSC7430's device's DDR_Vref supply and the DDR device's reference voltage.
- Generate the DDR_Vref from the V_{DD_IODDR} supply using a resistor divider with value of 1 kΩ and an accuracy of 1% or better.
- Use a decoupling capacitance of at least 0.1 µF on the supply in a manner similar to V_{DD_IODDR} and V_{SS} to ensure tracking of supply variations; however, the time constant of the resistor divider and decoupling capacitance should not exceed the nRESET assertion time after power on.
- V_{DD_IODDR} pins must not share vias. Use at least one via for each V_{DD_IODDR} pin. The extra inductance from sharing vias may cause bit errors in the DDR interface.

Routing recommendations:

- DDR_CK/DDR_CKn must be routed as a differential pair with a 100 Ω differential characteristic impedance.
- DDR_*x*DQS/DDR_*x*DQSn must be routed as a differential pair with a 100 Ω differential characteristic impedance.
- To minimize crosstalk, the characteristic impedance of the single-ended signals should be determined predominantly by the distance to the reference plane and not the distance to the neighboring traces.
- The crosstalk should be below –20 dB.
- If the DDR interface is not used, connect V_{DD_IODDR} and DDR_V_{REF} to ground. Leave all other DDR signals unconnected (floating). V_{DD_IODDR} can also be left floating; however, DDR_V_{REF} must also be left floating.

9.3.7 Thermal Diode External Connection

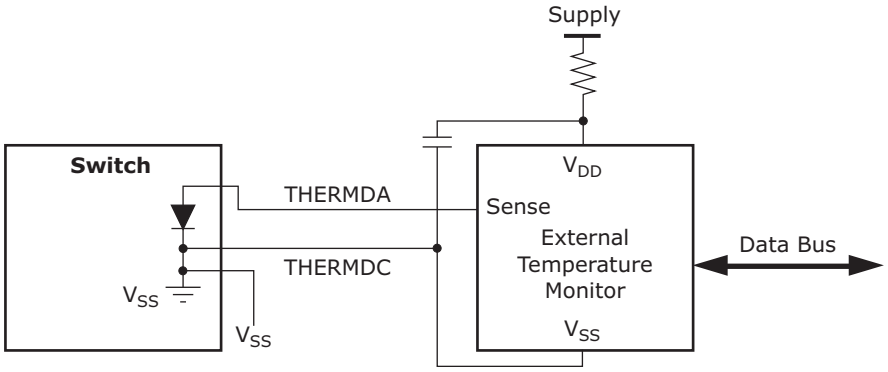
The internal on-die thermal diode can be used with an external temperature monitor to easily and accurately measure the junction temperature of this device.

The thermal diode is extremely sensitive to noise. To minimize the temperature measurement errors, follow these guidelines:

- Route the THERMDC and THERMDA signals as a differential pair with a differential impedance less than 100 Ω.
- Place the external temperature monitor as close as is possible to this device.
- Add a 47 Ω resistor in series with the external temperature monitor supply to filter noise.
- When using a grounded sensor circuit, place a de-coupling capacitor between the external temperature monitor supply pin and the THERMDC signal. For more information about using a grounded sensor circuit, see Figure 171, page 462.

Place the capacitor close to the external temperature sensor, as shown in the following illustration. Connect the external temperature monitor V_{SS} pin directly to the THERMDC pin, which has the connection to V_{SS}, as shown in the following illustration. Do not connect the external temperature monitor V_{SS} pin to the global V_{SS} plane.

Figure 194 **• External Temperature Monitor Connection**



10 Design Considerations

This section provides information about design considerations for the VSC7430 device.

OAM statistics may be incorrectly maintained during CPU Injection

OAM PDUs, carried in an Ethernet frame less than 149 bytes in size, are unaffected by this issue and are counted correctly. The following issue description only relates to the OAM PDUs carried in an Ethernet frame larger than 148 bytes.

When an OAM PDU is processed by a VOE (MEP) at the same time as the CPU injects any kind of traffic into the switch core, it can happen that the OAM statistics is incorrectly maintained by the processing VOE.

The issue has no impact on the OAM dataplane operation with peer MEPs. OAM-LM (loss measurements) will for example run correctly at all times. The issue only relates to maintaining statistics of local significance to the VOE, such as the number of selected and unselected OAM PDUs the processing VOE has transmitted and received.

All statistics in the register group VOP:VOE_STAT[VOE] are impacted by this issue.

Half-duplex is not working on the internal CuPHY interfaces

Half-duplex operation is possible on all SerDes interfaces. Half-duplex does not work for port 0 and 1 if the internal CuPHY is used.

Use external CuPHY interfaces for half-duplex operation on CuPHY interfaces. For fiber, SGMII operation half-duplex is working correctly.

11 Ordering Information

The VSC7430XMT package is a lead-free (Pb-free), 324-pin, plastic ball grid array (BGA) with a 19 mm × 19 mm body size, 1 mm pin pitch, and 1.8 mm maximum height.

The following table lists the ordering information for the VSC7430 device.

Table 329 • **Ordering Information**

Part Order Number	Description
VSC7430XMT	Lead-free, 324-pin, plastic BGA with a 19 mm × 19 mm body size, 1 mm pin pitch, and 1.8 mm maximum height