



Intel® Ethernet Controller E610

Datasheet

Ethernet Products Group (EPG)

PRODUCT FEATURES

Interfaces

- Dual/Single Port 10Gb / 5Gb / 2.5Gb / 1Gb / 100Mb Ethernet

SKU(s)

- E610-XAT2 - 2x10GBASE-T
- E610-XAIT2 - 2x10GBASE-T (Industrial Temp)
- E610-XAT - 1x10GBASE-T
- E610 IAT2 - 2x2.5GBASE-T
- E610IAIT2 - 2x2.5GBASE-T (Industrial Temp)

LAN Performance

- Based on Intel® Ethernet 500 Series architecture
- Intelligent Flow Direction: Receiver Side Scaling (RSS) and Intel® Ethernet Flow Director
- Tx/Rx IP, TCP, and UDP checksum offloading (IPv4, IPv6) capabilities (IPv4, IPv6), Tx TCP segmentation offload (IPv4, IPv6)

Security

- Latest in Firmware Resilience
- RSA3K / SHA384 Signed Firmware
- Integrated Hardware RoT with SPDM¹ firmware measurement support, with NIST 800-193 support

L2 Sorter and Traffic Classifiers

- Simple VEPA, VEB (Virtual Ethernet Bridge), VMDq for VMware Netqueue and Microsoft dVMQ
- Network Virtualization Stateless Offload: VXLAN, NVGRE (Supported on 10GbE SKUs only)
- SR-IOV (Single Root I/O Virtualization)

Management

- NC-SI
- MCTP over PCIe and SMBus/I²C
- PLDM over MCTP; PLDM Monitoring; PLDM Firmware Update; PLDM for RDE
- NET2BMC/OS2BMC

1. SPDM is at risk for initial production. A future stepping may be required to finalize the SPDM ROM image to support this feature.

Revision History

Revision	Date	Notes
1.2	June 15, 2026	- Updated "Dual/Single" for PRODUCT FEATURES - Interfaces - Added "Support 5G" row for Table 1-1, "Network Features" - Updated SDP[2] and SDP[1] for Table 2-7, "GPIO Pins" - Updated Table 11-7, "1000BASE-T Power Supply Specification" - Updated Table 12-5, "E610 NICs and LOMs"
1.1	November 25, 2025	- Added new E610 SKUs - Updated Table 11-7, "1000BASE-T Power Supply Specification" - Updated Table 12-5, "E610 NICs and LOMs"
1.0	March 1, 2025	Design Consideration and Guidelines (Intel Confidential)
0.9	October 2, 2024	Fifth preliminary draft (Intel confidential)
0.8	February 2, 2024	Fourth preliminary draft (Intel confidential)
0.7	June 28, 2023	Third preliminary draft (Intel confidential)
0.6	February 24, 2023	Second preliminary draft (Intel confidential)
0.5	October 4, 2022	Preliminary draft (Intel Confidential)

Chapter 1 Introduction	3
1.1 Scope	3
1.2 Product Overview	3
1.2.1 System Configurations	3
1.3 External Interfaces	4
1.3.1 PCIe Interface	4
1.3.2 Network Interfaces	4
1.3.3 Serial Flash Interface	5
1.3.4 SMBus Interface	5
1.3.5 NC-SI Interface	5
1.3.6 Software-Definable Pins (SDP) Interface (General-Purpose I/O)	5
1.3.7 LED Interface	5
1.4 Feature Summary	6
1.5 Conventions	11
1.5.1 Terminology and Acronyms	11
1.5.2 Byte Ordering	11
1.7 Architecture and Basic Operation	14
1.7.1 Transmit (Tx) Data Flow	14
1.7.2 Receive (Rx) Data Flow	15
Chapter 2 Pin Interface	17
2.1 Signal Type Definition	17
2.2 E610 Die Pin Assignments	18
2.2.1 PCIe Interface Pins	18
2.2.2 Ethernet Interface Pins	19
2.2.5 NC-SI Interface Pins	21
2.2.6 General Purpose I/O (GPIO) Pins	22
2.2.7 Miscellaneous Pins	24
2.2.10 PHY0 (10GBASE-T) Pins	27
2.2.11 PHY1 (10GBASE-T) Pins	28
2.2.14 PHY1 (10GBASE-T) Power Supplies	31
2.3.1 External Pull-Ups	32
2.4 Ball Out — Top View Through Package	33
Chapter 3 Interconnects	35
3.1 PCI Express (PCIe)	35
3.1.1 General Overview	35
3.1.2 Transaction Layer	36
3.1.3 Link Layer	45
3.1.4 Physical Layer	46
3.1.5 Error Events and Error Reporting	50
3.1.6 Performance and Statistics Counters	56
3.1.7 Data Object Exchange	65
3.2 Management Interfaces	67
3.2.1 SMBus	67
3.2.2 NC-SI (over RMII)	67
3.3 Non-Volatile Memory (NVM)	68
3.3.1 General Overview	68
3.3.2 External Flash	70
3.3.3 Shadow RAM	71
3.3.4 NVM Access Modes	73
3.3.5 NVM Update Flows	75
3.3.6 NVM Clients and Low-Level Interfaces	94
3.3.7 Flash Access Contention	95
3.3.8 NVM Access Procedures	96
3.3.9 NVM Authentication Procedure	99

3.3.10	NVM Access Admin Commands and Events	107
3.3.11	VPD Support.....	135
3.4	Configurable I/O Pins — Software-Definable Pins (SDPs)	137
3.4.1	I ² C Over SDP	138
3.5	General Purpose I/O (GPIO) and LEDs	142
3.5.1	LEDs	142
3.6	Network Interface	146
3.6.1	Overview	146
3.6.2	Internal MDIO Interface	146
3.6.3	Enabling the Network Link.....	147
3.6.4	Integrated Internal PHY (PHYSS) Functionality	148
3.6.5	Ethernet Flow Control (FC)	152
3.6.6	Inter Packet Gap (IPG) Control and Pacing.....	162
3.7	Link Management	164
3.7.1	Overview	164
3.7.2	Link Flows	166
3.7.8	DNL Engine	306
3.7.9	DNL Engine Commands.....	307
3.8	Link Topology	310
3.8.1	Overview	310
3.8.2	Link Topology Definition	311
3.8.3	Topology Structures	326
3.8.4	Link Topology Use Case Examples	334
3.8.5	Link Topology Adaptive NVM Features.....	337
Chapter 4	Initialization	403
4.1	E610 Power-Up	403
4.1.1	POR Flow	403
4.2	Reset Operation	404
4.2.1	Reset Sources/Reset Order.....	404
4.2.2	Reset in a PCI-IOV Environment.....	408
4.2.3	Firmware Reset.....	409
4.2.4	Reset Effects	409
4.2.5	Autoload Shadow RAM	412
4.3.1	Die Level Clock Interfaces	415
4.3.2	E610 Clocking Requirements.....	416
4.3.3	Clocking Bring-Up	417
4.3.4	Clocking Shutdown	417
4.4	Queue Disable	418
4.5	Device/Port/Function Configuration	418
4.5.1	General	418
4.5.2	Disable Through Strapping Pins.....	418
4.5.3	Port and Device Disable	420
4.5.4	Function Disable	420
4.5.5	Port Enable and Disable from BIOS (HII)	421
4.5.6	Event Flow for Enable/Disable Functions.....	422
4.6	Software Initialization and Diagnostics	423
4.6.1	Introduction	423
4.6.2	Power-Up State	423
4.6.4	Link Initialization.....	425
4.6.5	Initialization of Statistics	425
4.6.6	Interrupt Initialization.....	425
4.6.7	Receive Initialization.....	426
4.6.8	Transmit Initialization	428
4.6.9	Virtualization Initialization Flow	429
4.6.10	DCB Configuration.....	432

4.7	Access to Shared Resources	442
4.8	BIOS Init	442
4.8.2	Non-Persistent Configuration	443
4.8.4	Device State.....	452
4.9	Driver Load	452
4.9.1	Introduction	452
Chapter 5	Power Management and Delivery	453
5.1	Power Targets and Power Delivery	453
5.2	Power Management	453
5.2.1	Introduction to E610 Power States	453
5.2.2	Auxiliary Power Usage	454
5.2.3	PCIe Link Power Management	454
5.2.4	Power States	455
5.2.5	Timing of Power-State Transitions	459
5.3.1	PHY Power-Down State	463
5.3.2	PHY Power-Down via the PHY Register	464
5.3.3	Smart Power-Down (SPD)	464
5.3.4	Disable 10GBASE-T and/or 1000BASE-T Speeds.....	464
5.3.5	Energy Efficient Ethernet (EEE)	465
5.4.1	Advanced Power Management Wake-Up	468
5.4.2	ACPI Power Management Wake-Up.....	468
5.4.4	Wake-Up and Virtualization	473
5.5.1	LTR Algorithm.....	474
5.5.2	LTR Initialization Flow	474
Chapter 6	Non-Volatile Memory Map	477
6.1	NVM Organization	477
6.1.1	NVM Map High Level	477
6.1.4	NVM Integrity Checks by Software.....	493
6.1.5	Header of NVM Modules	494
6.1.6	Adaptive NVM Structures.....	498
6.2	PLDM Header	506
6.2.1	PLDM Header Section Summary Table.....	506
6.3	NVM Content	576
6.3.1	SPI Descriptor Section Summary Table	576
6.3.2	Init Module Section Summary Table.....	576
6.3.3	PFA Header Section Summary Table	603
6.3.4	PFA Features Module Section Summary Table	603
6.3.5	Feature Configuration Padding module Section Summary Table	604
6.3.6	PFA Immediate Values Module Section Summary Table	605
6.3.7	Immediate Fields Padding module Section Summary Table	606
6.3.8	PFA VPD Module Section Summary Table.....	607
6.3.9	PFA MNG Filter Section Summary Table.....	608
6.3.10	PFA PT configuration 0 Section Summary Table	609
6.3.11	PFA PT configuration 1 Section Summary Table	616
6.3.12	Original EETrack ID Section Summary Table	620
6.3.13	IBA capabilities Module Section Summary Table	621
6.3.14	PXE Setup Options Module Section Summary Table.....	623
6.3.15	PXE Configuration Customization Options Module Section Summary Table.....	625
6.3.16	PXE version Module Section Summary Table	628
6.3.17	VLAN Module Section Summary Table	629
6.3.18	Boot Configuration Block Section Summary Table	631
6.3.19	PBA header Section Summary Table	632
6.3.20	PBA Block Section Summary Table	633
6.3.21	PCIR Registers PFA Autoload Module Section Summary Table.....	635

6.3.22	POR Registers PFA Autoload Module Section Summary Table.....	654
6.3.23	MinSrev Section Summary Table	659
6.3.24	PF MAC Address Section Summary Table	661
6.3.25	MNG MAC Address Section Summary Table.....	664
6.3.26	FW Logging Defaults Section Summary Table.....	666
6.3.27	MD Link Topology Section Summary Table.....	675
6.3.28	RDE Module Section Summary Table.....	677
6.3.29	Identical content as PLDM header ComponentImageSetVersionString Section Summary Table ...	687
6.3.30	Software Checksum Module Section Summary Table	691
6.3.31	Link Default Override Mask Section Summary Table	692
6.3.32	RDE Ethernet MTU Section Summary Table.....	697
6.3.33	HII Port Disable By Function Section Summary Table	698
6.3.34	NetlistMinSrev Section Summary Table	699
6.3.35	OEM RoT Key Revocation Module Section Summary Table.....	700
6.3.36	OEM RoT Key Revoke ISK Object CSS Header Section Summary Table	701
6.3.37	OEM RoT Key Revoke ISK Object Key and Signature Section Summary Table	707
6.3.38	OEM RoT Key Revoke ISK Object Auth Header Section Summary Table	709
6.3.39	OEM RoT Key Revoke ISK Object Section Summary Table.....	711
6.3.40	OEM RoT Key Revocation Module CSS Header Section Summary Table	712
6.3.41	OEM RoT Key Revocation Module Key and Signature Section Summary Table	718
6.3.42	OEM RoT Key Revocation Module Auth Header Section Summary Table	720
6.3.43	OEM RoT Key Revocation Module Data Section Summary Table	722
6.3.44	Padding module Section Summary Table	723
6.3.45	PCIR Type 1/2 Section Summary Table	724
6.3.46	Sideband Bus Autoload Section Summary Table	727
6.3.47	EMP SR Settings Module Header Section Summary Table.....	729
6.3.48	GFID Module Section Summary Table	737
6.3.49	Manageability Module Header Section Summary Table.....	744
6.3.50	Sideband Configuration Structure Section Summary Table	749
6.3.51	OEM Section Section Summary Table	754
6.3.52	Auto Generated Pointers Module Section Summary Table.....	755
6.3.53	MAC Reset 0 Registers Autoload Module Section Summary Table	759
6.3.54	Master Reset 0 Registers Autoload Module Section Summary Table.....	762
6.3.55	OEM Port Config 0 Section Summary Table	767
6.3.56	NVM Image CSS Header Section Summary Table.....	778
6.3.57	NVM Key and Signature Section Summary Table	784
6.3.58	NVM Image Auth Header Section Summary Table.....	786
6.3.59	Padding section Section Summary Table.....	787
6.3.60	SR1 - should be copy of Shadow RAM: Section clone.....	787
6.3.61	ML CSS Header Section Summary Table.....	788
6.3.62	ML Key and Signature Section Summary Table.....	794
6.3.63	ML Auth Header Section Summary Table	796
6.3.64	Extended ML Header Section Summary Table.....	798
6.3.65	ML Image Section Summary Table.....	804
6.3.66	Analog PHY pre PLL Configuration Section Summary Table.....	804
6.3.67	CSR Protected List Section Summary Table.....	805
6.3.68	PCIe Analog Module Section Summary Table.....	835
6.3.69	PCIR Registers PFA Autoload Module Section Summary Table.....	836
6.3.70	POR Registers Autoload Module Section Summary Table.....	840
6.3.71	PCIR_PFA Autoload Allowlist Module Section Summary Table.....	841
6.3.72	POR_PFA Autoload Allowlist Module Section Summary Table	860
6.3.73	MAC Reset Registers Autoload Module Section Summary Table.....	866
6.3.74	Master Reset Registers Autoload Module Section Summary Table.....	870
6.3.75	NVM ISK Object CSS Header Section Summary Table.....	877
6.3.76	NVM ISK Object Key and Signature Section Summary Table.....	883

6.3.77	NVM ISK Object Auth Header Section Summary Table	885
6.3.78	NVM ISK Object Section Summary Table	887
6.3.79	Signature RoT Key Revocation Module CSS Header Section Summary Table	888
6.3.80	Signature RoT Key Revocation Module Key and Signature Section Summary Table	894
6.3.81	Signature RoT Key Revocation Module Auth Header Section Summary Table	896
6.3.82	Signature RoT Key Revocation Module Data Section Summary Table	898
6.3.83	Recovery FW CSS Header Section Summary Table	899
6.3.84	Recovery FW Key and Signature Section Summary Table	905
6.3.85	Recovery FW Auth Header Section Summary Table	907
6.3.86	Image Raw Data Section Summary Table	908
6.3.87	EMP Global Module Section Summary Table	909
6.3.88	EMP Settings Module Header Section Summary Table	913
6.3.89	Allow list Section Summary Table	916
6.3.90	Configuration Metadata Section Summary Table	916
6.3.91	DL Scripts Section Summary Table	916
6.3.92	Analog PHY Configuration Section Summary Table	917
6.3.93	EMP Image Section Summary Table	918
6.3.94	RDE Dictionaries Section Summary Table	918
6.3.95	External Topology Device Image 0 Section Summary Table	919
6.3.96	External Topology Device Image 1 Section Summary Table	921
6.3.97	External Topology Device Image 2 Section Summary Table	923
6.3.98	External Topology Device Image 3 Section Summary Table	925
6.3.99	NVM Hash Module Section Summary Table	927
6.3.100	NVM Provisioning Area Section Summary Table	971
6.3.101	OROM Section Summary Table	971
6.3.102	OROM ISK Object CSS Header Section Summary Table	972
6.3.103	OROM ISK Object Key and Signature Section Summary Table	978
6.3.104	OROM ISK Object Auth Header Section Summary Table	980
6.3.105	OROM ISK Object Section Summary Table	982
6.3.106	OROM CSS Header Section Summary Table	983
6.3.107	OROM Key and Signature Section Summary Table	989
6.3.108	OROM Auth Header Section Summary Table	991
6.3.109	OROM Provisioning Area Section Summary Table	992
6.3.110	Link Topology Netlist Section Summary Table	993
6.3.111	Link Topology Netlist ISK Object CSS Header Section Summary Table	994
6.3.112	Link Topology Netlist ISK Object Key and Signature Section Summary Table	1000
6.3.113	Link Topology Netlist ISK Object Auth Header Section Summary Table	1002
6.3.114	Link Topology Netlist ISK Object Section Summary Table	1004
6.3.115	Link Topology Netlist CSS Header Section Summary Table	1005
6.3.116	Link Topology Netlist Key and Signature Section Summary Table	1011
6.3.117	Link Topology Netlist Auth Header Section Summary Table	1013
6.3.118	TLV extension Provisioning Area Section Summary Table	1015
6.3.119	Link Topology Scratch Pad Area Section Summary Table	1016
6.3.120	FW Scratch Pad Area Section Summary Table	1016
6.3.121	Factory Settings Header Section Summary Table	1017
6.3.122	Factory Settings Area Section Summary Table	1018
6.3.123	Guarded Zone Section Summary Table	1019
Chapter 7	Inline Functions	1021
7.1	Receive Functionality	1021
7.1.1	MAC Layer - Receive	1021
7.1.2	Packet Filtering	1022
7.1.3	Rx Queues Assignment	1026
7.1.4	Receive Data Storage in System Memory	1051
7.1.5	Receive Descriptors	1052
7.1.6	Receive Offloads	1064

7.1.7	Receive Statistics	1069
7.2	Transmit Functionality	1071
7.2.1	Packet Transmission	1071
7.2.2	Transmit Contexts	1080
7.2.3	Transmit Descriptors	1080
7.2.4	TCP and UDP Segmentation	1094
7.2.5	Transmit Checksum Offloading in Non-Segmentation Mode	1101
7.2.6	Transmit Statistics	1103
7.3	Interrupts	1105
7.3.1	Interrupt Registers	1105
7.3.2	Interrupt Moderation	1108
7.3.3	TCP Timer Interrupt	1109
7.3.4	Mapping of Interrupt Causes	1109
7.4	802.1q VLAN Support	1116
7.4.1	802.1q VLAN Packet Format	1116
7.4.2	802.1q Tagged Frames	1116
7.4.3	Transmitting and Receiving 802.1q Packets	1117
7.4.4	802.1q VLAN Packet Filtering	1117
7.4.5	Double VLAN and Single VLAN Support	1118
7.4.6	E-tag and VLAN	1120
7.5	TLP Processing Hints (TPH)	1122
7.5.1	Steering Tag and Processing Hint Programming	1122
7.6	Data Center Bridging (DCB)	1123
7.6.1	Overview	1123
7.6.2	Transmit-Side Capabilities	1125
7.6.3	Receive-Side Capabilities	1137
7.7	Precise Timing Protocol (PTP)	1141
7.7.1	Overview	1141
7.7.2	Flow and Hardware/Software Responsibilities	1141
7.7.3	Hardware PTP Elements	1144
7.7.4	Hardware PTP Elements	1146
7.7.5	PTP Interrupts	1148
7.7.6	PTP Packet Structure	1149
7.8	Virtualization	1153
7.8.1	Overview	1153
7.8.2	PCI-SIG SR-IOV Support	1157
7.8.3	Packet Switching	1167
7.8.4	Security Features	1177
7.9	Tunneling Support	1182
7.10	Reliability	1183
7.10.1	Memory Integrity Protection	1183
7.10.2	PCIe Error Handling	1183
Chapter 8	Programming Interface	1185
8.1	General	1185
8.1.1	Memory-Mapped Access	1185
8.1.2	I/O-Mapped Access	1186
8.1.3	Configuration Access to Internal Registers and Memories	1187
8.1.5	VF Registers Allocated per Queue	1189
8.1.6	Non-Queue VF Registers	1190
8.2	Device Registers - PF	1191
8.2.1	BAR0 Registers Summary	1191
8.2.2	Detailed Register Description - PF BAR0	1208
8.3	Device Registers - VF	1382
8.3.1	VF Registers Mapping in the PF Space	1382
8.3.2	BAR0 Registers Summary	1382

8.3.3	Detailed Register Description - VF BAR0	1385
8.3.4	BAR4 Registers Summary.....	1395
8.3.5	Detailed Register Description - VF BAR4	1395
Chapter 9	PCIe Programming Interface	1399
9.1	Overview	1399
9.2.1	PCIe Configuration Space Summary.....	1401
9.2.2	Mandatory PCI Configuration Registers.....	1403
9.2.3	PCI Capabilities.....	1410
9.2.4	PCIe Extended Configuration Space	1434
9.2.5	CSR Access Via Configuration Address Space	1462
9.3	Virtual Functions Configuration Space	1463
9.3.2	PCI Capabilities.....	1466
9.3.3	PCIe Extended Capabilities	1468
Chapter 10	System Manageability	1471
10.1	Features	1471
10.2	Pass-Through (PT) Functionality	1472
10.2.1	Supported Topologies	1473
10.2.2	Pass-Through Packet Routing.....	1473
10.3	Components of the Sideband Interface	1473
10.3.1	Physical Layer.....	1473
10.3.2	Logical layer.....	1475
10.4	Packet Filtering	1481
10.4.1	Manageability Receive Filtering	1482
10.4.2	L2 Filters.....	1483
10.4.3	L3/L4 Filtering	1483
10.4.4	Flexible 128 Byte Filter	1485
10.4.5	Configuring Manageability Filters.....	1486
10.4.6	Filtering Programming Interfaces.....	1490
10.4.7	Possible Configurations	1490
10.4.8	Determining Manageability MAC Address	1492
10.5.1	Overview	1493
10.5.2	Filtering	1494
10.5.3	Blocking of Network to BMC Flow.....	1495
10.5.4	OS2BMC and Flow Control	1495
10.5.5	Statistics.....	1496
10.5.6	OS-to-BMC Enablement	1496
10.6.1	Overview	1497
10.6.2	NC-SI Standard Support	1501
10.6.3	External Link Control via NC-SI	1505
10.6.4	NC-SI Mode — Intel-Specific Commands	1507
10.6.5	Basic NC-SI Workflows.....	1558
10.6.6	Asynchronous Event Notifications (AENs).....	1562
10.6.7	Querying Active Parameters.....	1562
10.6.8	Resets	1562
10.6.9	Advanced Workflows.....	1563
10.7.1	Management Component Transport Protocol (MCTP) Overview	1566
10.7.3	MCTP Over SMBus.....	1572
10.7.4	NC-SI over MCTP	1573
10.7.5	PLDM over MCTP	1577
10.7.6	OEM Commands over MCTP.....	1577
10.7.7	MCTP Programming	1578
10.8	PLDM Support	1584
10.8.1	PLDM Base Implementation	1584
10.8.2	PLDM Monitoring and Control Support	1589

10.8.3	PLDM Monitoring and Control - Generic Structures	1611
10.8.4	PLDM Firmware Update Commands	1646
10.8.5	PLDM Firmware Update Flow	1652
10.8.6	RDE Support	1657
10.9.1	Support for FAN_ON_AUX Pin	1701
10.10.1	SPDM Enablement	1702
10.10.2	Secure Session Limit	1702
10.10.3	Measurement Algorithm	1702
10.11	Firmware Host Interface	1717
10.11.1	Introduction	1717
10.11.2	CSR-Based Firmware Admin Commands for Driver & Tools	1717
10.11.3	Legacy Manageability Host Interface	1744
10.12	Host Isolate Support	1752
Chapter 11 Electrical/Mechanical Specification		1755
11.1	Introduction	1755
11.2	Operating Conditions	1755
11.2.1	Absolute Maximum Ratings	1755
11.3	Power Delivery	1756
11.3.1	Power Supply Specification	1756
11.4.1	MAX Power (TDP)	1762
11.5.1	Digital I/O DC Specifications	1764
11.5.2	Digital I/F AC Specifications	1765
11.5.3	PCIe Interface AC/DC Specification	1771
11.5.4	Network Interface AC/DC Specification	1771
11.5.5	Reference Clock Specification	1771
11.6.1	Mechanical Configuration	1777
11.6.2	Heat Sink Mechanical Load Limits	1777
11.6.3	Thermal Resistance Parameters	1777
11.6.4	Electrical	1777
11.7	Package Mechanical Drawings	1778
11.8.1	Flash	1781
Chapter 12 Design Considerations and Guidelines		1783
12.1	Connecting the PCIe Interface	1783
12.1.1	Link Width Configuration	1783
12.1.2	Polarity Inversion and Lane Reversal	1783
12.1.3	PCIe Reference Clock	1784
12.1.4	Bias Resistor	1784
12.1.5	Miscellaneous PCIe Signals	1784
12.1.6	PCIe Layout Recommendations	1784
12.2	Connecting the 10GBASE-T MDI Interfaces	1784
12.2.1	MDI Circuit Guidelines	1785
12.2.2	Magnetics Module	1785
12.2.3	MDI Layout Guidance	1785
12.2.5	Center Tap Connection Via Capacitors to Ground	1792
12.4	Connecting the Flash Interface	1794
12.4.1	Connecting the Flash	1794
12.4.2	Supported Flash Devices	1794
12.5	Connecting Manageability Interfaces	1795
12.5.1	Connecting the SMBus Interface	1795
12.5.2	Connecting the NC-SI Interface	1795
12.5.3	Layout Requirements	1797
12.6	Connecting the Software-Definable Pins (SDPs)	1798
12.7	Connecting the Light Emitting Diodes (LEDs)	1798
12.8	Connecting Miscellaneous Signals	1799

12.8.1	LAN Disable	1799
12.8.2	Event Flow for Enable/Disable Functions (HII)	1801
12.9	Connecting the JTAG Port	1801
12.10	Power On Reset (POR)	1801
12.11	Clock Source Design Considerations	1802
12.11.1	Quartz Crystal	1802
12.11.2	Vibrational Mode	1802
12.11.3	Frequency Tolerance	1802
12.11.4	Temperature Stability and Environmental Requirements	1802
12.11.5	Calibration Mode	1803
12.11.6	Reference Crystal Circuit	1803
12.11.7	Crystal Load Capacitance	1803
12.11.8	Shunt Capacitance	1804
12.11.9	Equivalent Series Resistance (ESR)	1804
12.11.10	Driver Level	1804
12.11.11	Aging	1804
12.11.12	Reference Crystal	1804
12.11.14	Circuit Board	1805
12.11.15	Temperature Changes	1805
12.12.1	Board Stack-Up Example	1806
12.12.2	Customer Reference Board Stack-Up Example	1807
12.12.3	Via Usage	1809
12.12.4	Reference Planes	1811
12.12.5	Reducing Circuit Inductance	1812
12.12.6	Signal Isolation	1813
12.12.7	Traces for Decoupling Capacitors	1813
12.12.9	Recommended Simulations	1819
12.13	Bill Of Material (BOM)	1820
Chapter 13 Thermal Design Considerations		1821
13.1	Introduction	1821
13.2	Measuring the Thermal Conditions	1821
13.3	Thermal Considerations	1821
13.4	Importance of Thermal Management	1822
13.5	Packing Terminology	1822
13.6	Thermal Specifications	1823
13.7	Package Mechanical Attributes	1824
Chapter 14 Reliability, Diagnostics and Testability		1825
14.1	Reliability	1825
14.1.1	ECC Support and ECC Error Flow	1825
14.2	Link Loopback Operations	1826
14.2.1	Firmware Logging Overview	1826
14.2.2	Health Status Commands	1829
14.3	Firmware Recovery Mode	1835
14.3.1	Overview	1835
14.3.2	Recovery Flows	1836
14.3.3	Operation Mode Software Identification	1838
Chapter 15 Glossary and Acronyms		1839



Chapter 1 Introduction

1.1 Scope

This document describes the external architecture (including device operation, pin descriptions, register definitions, etc.) for the E610, a dual port 1/10 Gigabit Ethernet Controller.

This document is intended as a reference for logical design group, architecture validation, firmware development, software device driver developers, board designers, test engineers, or anyone else who might need specific technical or programming information about the E610.

1.2 Product Overview

The E610 is a derivative of the X550. Many features of its predecessor remain intact; however, some have been removed or modified as well as new features introduced. Notably, security has been significantly improved to align with current state of the art security standards and trends. This includes, Hardware RoT, firmware authentication on update & load, fuse based SVN, key revocation, and others.

1.2.1 System Configurations

The E610 is targeted for system configurations such as rack mounted, pedestal servers or workstations, where it can be implemented used as an add-on NIC or LAN on Motherboard (LOM), or purchased from Intel as a standard PCIe adapter card.

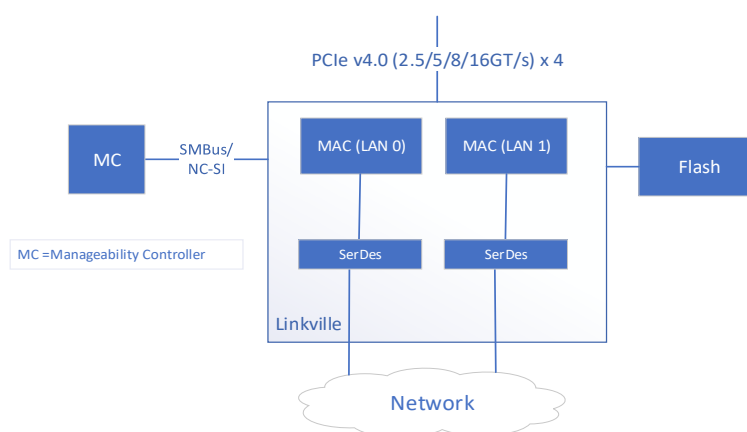


Figure 1-1. Typical Rack/Pedestal System Configuration

1.3 External Interfaces

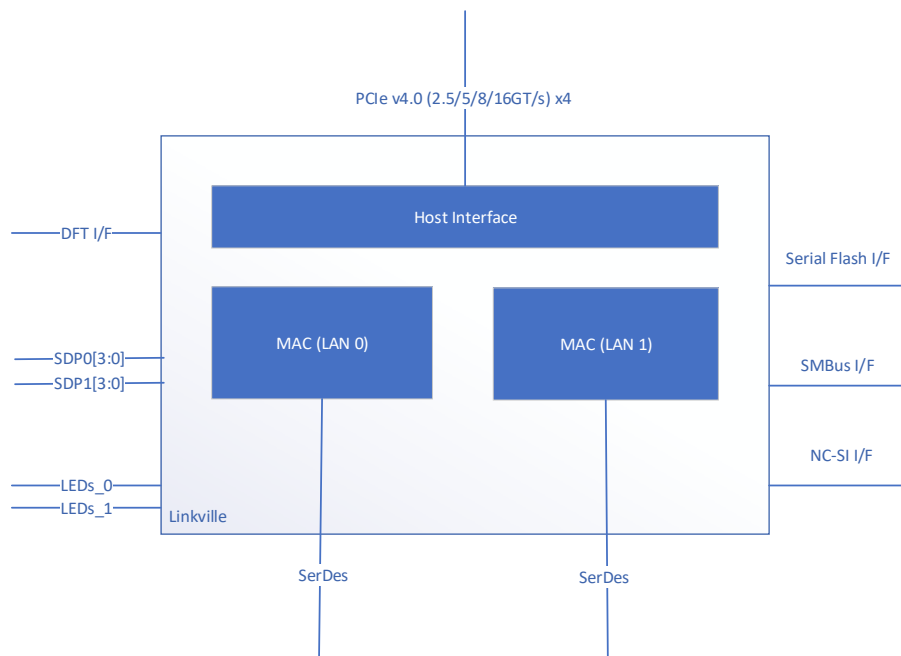


Figure 1-2. E610 External Interfaces Diagram

1.3.1 PCIe Interface

The E610 supports PCIe v4.0 (2.5 GT/s, 5 GT/s, 8 GT/s or 16GT/s). See [Section 2.2.1](#) for full pin description and [Section 11.5.3](#) for interface timing characteristics.

1.3.2 Network Interfaces

Two independent BASE-T interfaces are used to connect the two E610 ports to external devices. Each interface can operate at any of the following speeds:

- 100BASE-TX
- 1000BASE-T
- 2500BASE-T
- 5GBASE-T
- 10GBASE-T
- Refer to [Section 2.2.2](#) for full-pin descriptions. For the timing characteristics of those interfaces, refer to the relevant external specifications listed in [Section 11.5.4](#).

1.3.3 Serial Flash Interface

The E610 uses an external SPI serial interface to a Flash device, also referred to as Non-Volatile Memory (NVM). The E610 supports serial Flash devices with up to 32 MB of memory. The data interface is a 4-wire interface with clock speed of up to 50 MHz.

1.3.4 SMBus Interface

SMBus is an optional interface for pass-through and/or configuration traffic between an external Manageability Controller (MC) and the E610.

The E610's SMBus interface supports a standard SMBus, up to a frequency of 400 KHz. Refer to [Section 2.2.4](#) for full-pin descriptions and [Section 10.3.1.1](#) for timing characteristics of this interface.

1.3.5 NC-SI Interface

NC-SI is an optional interface for pass-through traffic to and from an MC. The E610 meets the NC-SI version 1.1 specification.

Refer to [Section 2.2.5](#) for the pin descriptions, and [Section 10.6](#) for NC-SI programming.

1.3.6 Software-Definable Pins (SDP) Interface (General-Purpose I/O)

The E610 has four SDP pins per port that can be used for miscellaneous hardware or software-controllable purposes. These pins can each be individually configured to act as either input or output pins. Via the SDP pins, the E610 can support IEEE1588 auxiliary device connections and other functionality. For more details on the SDPs see [Section 3.4](#) and the ESDP register.

1.3.7 LED Interface

The E610 implements four output drivers intended for driving external LED circuits per port. Each of the four LED outputs can be individually configured to select the particular event, state, or activity, which is indicated on that output. In addition, each LED can be individually configured for output polarity as well as for blinking versus non-blinking (steady-state) indications.

The configuration for LED outputs is specified via the LEDCTL register. In addition, the hardware-default configuration for all LED outputs can be specified via an NVM field (see [Section 6.3.74.4](#) and [Section 6.3.74.5](#)), thereby supporting LED displays configured to a particular OEM preference. For more details on the LEDs see [Section 3.5](#).

1.4 Feature Summary

Table 1-1 to Table 1-7 list the E610's features in comparison to previous dual-port 10 GbE Ethernet controllers.

Table 1-1. Network Features

Feature	E610	X552/X553	82599	X550
Compliant with the 10 GbE and 1 GbE Ethernet/802.3Clause 70, 71 (KX/KX4) Specification	Y (KX only)	N/A	Y	N
Compliant with the 10 GbE 802.3Clause 72 (KR) Specification	Y	N/A	Y	N
Compliant with XFI/SFI Interface	SFI	N/A	Y	N
Compliant with the 1000BASE-BX Specification	N	N/A	Y	N
SerDes Interface for External PHY Connection or System Interconnect	Y	N/A	Y	N
SGMII Interface	Y	N/A	Y (100Mb/s and 1GbE only)	N
Support for 10G/1G/100M/10M speeds	Y (no support for 10M)	Y	Y	Y
Support 5G	Y	N	N	N
Support 2.5G	Y	Y	N	N
Support 1G	Y	Y	Y	Y
Support 100M	Y	Y	Y	Y
Support 10M	N	Y	N	N
Full-Duplex Operation at all Supported Speeds	Y	Y	Y	Y
Half-Duplex at 100 Mb/s Operation	N	N	N	N
10 GbE/1 GbE/100 Mb/s Copper PHYs Integrated On-Chip	MCP only	N	N	Y
802.3az Energy Efficient Ethernet (EEE) Support	Y	Y	N	Y
Support Jumbo Frames of up to 9.5-15.5 KB	Y ¹	Y ¹	Y ¹	Y ¹
MDIO Interface Clause 45	Y	Y	Y	Y (internally)
Flow Control Support: Send/Receive Pause Frames and Receive FIFO Thresholds	Y	Y	Y	Y
Statistics for Management and RMON	Y	Y	Y	Y
802.1q VLAN Support	Y	Y	Y	Y
Double VLAN	Y	Y	Y	Y

1. The E610 supports 9.5 KB jumbo packets. The X550 support full-size 15.5 KB jumbo packets while in a basic mode of operation. Packets to/from the MC longer than 2 KB are filtered out.

Table 1-2. Host Interface Features

Feature	E610	X552/X553	82599	X550
PCIe* Version (Speed)	PCIe v4.0 (16/8/5/2.5 GT/s)	N/A	PCIe v2.0 (5/2.5 GT/s)	PCIe v3.0 (8/5/2.5 GT/s)
Number of Lanes	x1, x2, x4	N/A	x1, x2, x4, x8	x1, x2, x4, x8 (Gen2 only)
IOSF Version Speed	Internal (IOSF 1.1 256b @ 500MHz)	IOSF 1.1 256b @ 300MHz	N/A	N/A
64-bit Address Support for Systems Using More Than 4 GB of Physical Memory	Y	Y	Y	Y
Outstanding Requests for Tx Data Buffers	16	16	16	16
Outstanding Requests for Tx Descriptors	8	8	8	8
Outstanding Requests for Rx Descriptors	8	8	8	8
Credits for P-H/P-D/NP-H/NP-D (shared for the two ports)	4/255/4/255 ¹	4/255/4/255 ²	16/16/4/4	16/16/4/4
Credits for C-H/C-D (shared for the two ports)	4/255	4/255		
Max Payload Size Supported	512 Bytes	64 thru 512 Bytes	512 Bytes	512 Bytes
Max Request Size Supported	2 KB	64 thru 2K Bytes	2 KB	2 KB
Link Layer Retry Buffer Size (shared for the two ports)	3.4 KB	N/A	3.4 KB	3.4 KB
Vital Product Data (VPD)	Y	Y	Y	Y
End to End CRC (ECRC)	Y	Y ³	Y	Y
TLP Processing Hints (TPH)	Y	N	N	Y
Latency Tolerance Reporting (LTR)	Y	N	N	Y
ID-Based Ordering (IDO)	Y	Y	N	Y
Access Control Services (ACS)	Y	Y	N	Y
ASPM Optional Compliance Capability	Y	Y	N	Y
PCIe Functions Off Via Pins, While LAN Ports Are On	Y	Y	N	Y

1. Each data credit accounts for 16B.

2. Each data credit accounts for 16B.

3. ECRC support is provided only when IP is integrated into a PCIe device.

Table 1-3. Miscellaneous Features

Feature	E610	X552/X553	82599	X550
Serial Flash Interface	Y	Y	Y	Y
4-wire SPI EEPROM Interface	Y	N	Y	N
Configurable LED Operation for Software or OEM Customization of LED Displays	Y	Y	Y	Y
Protected NVM Space for Private Configuration	Y	Y	Y	Y
Device Disable Capability	Y	Y	Y	Y
Package Size	TBD	N/A	25 mm x 25 mm	25 mm x 25 mm
Watchdog Timer	Y	Y	Y	Y

Table 1-3. Miscellaneous Features [continued]

Feature	E610	X552/X553	82599	X550
Time Sync (IEEE 1588)	Y	Y	Y	Y ¹
NVM authentication on update	Y	Y	Y	Y
NVM authentication on read	Y	N	N	N
Firmware authentication RSA3K-PKCSv1.5	Y	N	N	N
Key validation: SHA384	Y	N	N	N
SPDM	Y	N	N	N
Secure MCTP	Y	N	N	N
Fused based SVN (Security Version Number)	Y	N	N	N
Key revocation	Y	N	N	N

1. Time sync not supported at 100 Mb/s link speed.

Table 1-4. LAN Functions Features

Feature	E610	X552/X553	82599	X550
Programmable Host Memory Receive Buffers	Y	Y	Y	Y
Descriptor Ring Management Hardware for Transmit and Receive	Y	Y	Y	Y
ACPI Register Set and Power Down Functionality Supporting D0 and D3 States	Y	Y	Y	Y
Integrated MACsec, 801.2AE Security Engines: AES-GCM 128-bit; Encryption + Authentication; One SC x 2 SA Per Port. Replay Protection with Zero Window	N	N	Y	Y
Integrated IPsec Security Engines: AES-GCM 128-bit; AH or ESP encapsulation; IPv4 and IPv6 (no option or extended headers)	N	N	1024 SA / port	1024 SA / port
Software-Controlled Global Reset Bit (Resets Everything Except the Configuration Registers)	Y	Y	Y	Y
Software-Definable Pins (SDP) (per port)	4	2	8	4
SDP Pins Can Be Configured as General Purpose Interrupts	Y	Y	Y	Y
Wake on LAN (WoL)	Y	Y	Y	Y
IPv6 Wake-up Filters	Y	Y	Y	Y
Configurable (through NVM) Wake-up Flexible Filters	Y	Y	Y	Y
Default Configuration by NVM for all LEDs for Pre-Driver Functionality	Y	Y	Y	Y
LAN Function Disable Capability	Y	Y ¹	Y	Y
Programmable Memory Transmit Buffers	160 KB / port	160 KB / port	160 KB / port	160 KB / port
Programmable Memory Receive Buffers	384 KB / port	384 KB / port	512 KB / port	384 KB / port

1. Unlike SGVL LAN Function Disable cannot be done using SDPs

Table 1-5. LAN Performance Features

Feature	E610	X552/X553	82599	X550
TCP/UDP Segmentation Offload	256 KB in all modes	256 KB in all modes	256 KB in all modes	256 KB in all modes
TSO Interleaving for Reduced Latency	Y	Y	Y	Y
TCP Receive Side Coalescing (RSC)	N	32 flows / port	32 flows / port	32 flows / port
Data Center Bridging (DCB), IEEE Compliance to Enhanced Transmission Selection (ETS) - 802.1Qaz Priority-based Flow Control (PFC) - 802.1Qbb	Y (up to 8) Y (up to 8)	Y (up to 8) Y (up to 8)	Y (up to 8) Y (up to 8)	Y (up to 8) Y (up to 8)
Rate Limit VM Tx Traffic per TC (i.e. per TxQ)	Y	Y	Y	Y
IPv6 Support for IP/TCP and IP/UDP Receive Checksum Off load	Y	Y	Y	Y
Fragmented UDP Checksum Off load for Packet Reassembly	Y	Y	Y	Y
FCoE Tx / Rx CRC Off load	N	N	Y	Y
FCoE Transmit Segmentation	N/A	N/A	256 KB	256 KB
FCoE Coalescing and Direct Data Placement	N/A	N/A	512 outstanding Read — Write requests / port	2048 outstanding Read — Write requests / port
Message Signaled Interrupts (MSI)	Y	Y	Y	Y
Message Signaled Interrupts (MSI-X)	Y	Y	Y	Y
Interrupt Throttling Control to Limit Maximum Interrupt Rate and Improve CPU Use	Y	Y	Y	Y
Rx Packet Split Header	Y	Y	Y	Y
Multiple Rx Queues (RSS)	Y (multiple modes)	Y (multiple modes)	Y (multiple modes)	Y (multiple modes)
Flow Director Filters: up to 32 KB Flows by Hash Filters or up to 8 KB Perfect Match Filters	Y	Y	Y	Y
Number of Rx Queues (per port)	128	128	128	128
Number of Tx Queues (per port)	128	128	128	128
Low Latency Interrupts (LLI)	N	N	Yes to all	N
DCA Support	N	N		N
TCP Timer Interrupts	Y	Y		Y
No Snoop	N	N		N
Relax Ordering	Y	Y		Y
DMA Coalescing	N	Y	N	Y

Table 1-6. Virtualization Features

Feature	E610	X552/X553	82599	X550
Support for Virtual Machine Device Queues (VMDq1 and Next Generation VMDq)	64	64	64	64
L2 Ethernet MAC Address Filters (Unicast and Multicast)	128	128	128	128
L2 VLAN Filters	64	64	64	64
PCI-SIG SR IOV	Y	Y	Y	Y

Table 1-6. Virtualization Features [continued]

Feature	E610	X552/X553	82599	X550
Multicast and Broadcast Packet Replication	Y	Y	Y	Y
Packet Mirroring	Y	Y	Y	Y
Packet Loopback	Y	Y	Y	Y
RSS Replication	Y	Y	N	Y
Traffic Shaping	Y	Y	Y	Y
Anti Spoof	MAC, VLAN, Ethertype	MAC, VLAN, Ethertype	MAC, VLAN	MAC, VLAN, Ethertype
Malicious Driver Protection	Y	Y	N	Y
Forwarding Modes	MAC, VLAN E-tag	MAC, VLAN E-tag	MAC, VLAN	MAC, VLAN E-tag
VEB Support	Y	Y	Y	Y
VEPA Support	Y	Y	N	Y
E-tag filtering support	Y	Y	N	Y
Tunneling protocols NVGRE, VXLAN	Y	Y	N	Y
Tunneling protocol GENEVE	Y	Y	N	N

Table 1-7. Manageability Features

Feature	E610	X552/X553	82599	X550
Advanced Pass Through-compatible Management Packet Transmit/Receive Support	Y	Y	Y	Y
SMBus Interface to an External MC	Y	Y	Y	Y
NC-SI Interface to an External MC	Y	Y	Y	Y
L2 Address Filters	4	4	4	4
VLAN L2 Filters	8	8	8	8
Flex L3 Port Filters	16	16	16	16
Flexible TCO Filters	1	1	4	1
L3 Address Filters (IPv4)	4	4	4	4
L3 Address Filters (IPv6)	4	4	4	4
Host-Based Application-to-BMC Network Communication Patch (OS2BMC)	Y	Y	N	Y
Flexible MAC Address	Y	Y	N	Y
MC Inventory of LOM Device Information	Y	Y	N	Y
iSCSI Boot Configuration Parameters via MC	Y	Y	N	Y
MC Monitoring	Y	Y	N	Y
NC-SI to MC	Y	Y	N	Y
NC-SI Arbitration	Y	Y	N	Y
MCTP over SMBus (pass through and control)	Y	Y	N	Y
MCTP over PCIe (pass through and control)	Y	N/A	N	Y
MCTP over IOSF (pass through and control)	N/A	Y	N/A	N/A

Table 1-7. Manageability Features [continued]

Feature	E610	X552/X553	82599	X550
NC-SI Package ID Via SDP Pins	Y	Y	N	Y
NC-SI Flow Control	Y	Y	N	Y
Legacy SMBus pass through	N	Y	Y	Y
PDLM Monitoring and Control	Y	N	N	N
PLDM Firmware Update	Y	N	N	N
PLDM RDE support	Y	N	N	N
SPDM support	Y	N	N	N
Proxying	N	N	N	N

1.5 Conventions

1.5.1 Terminology and Acronyms

See [Chapter 15, “Glossary and Acronyms”](#).

1.5.2 Byte Ordering

This section defines the organization of registers and memory transfers, as it relates to information carried over the network:

- Any register defined in Big Endian notation can be transferred as is to/from Tx and Rx buffers in the host memory. Big Endian notation is also referred to as being in network order or ordering.
- Any register defined in Little Endian notation must be swapped before it is transferred to/from Tx and Rx buffers in the host memory. Registers in Little Endian order are referred to being in host order or ordering.

Tx and Rx buffers are defined as being in network ordering; they are transferred as is over the network.

Note: Registers not transferred on the wire are defined in Little Endian notation. Registers transferred on the wire are defined in Big Endian notation, unless specified differently.

1.6 References

The E610 implements features from the following specifications:

IEEE Specifications:

- 10GBASE-T as per the IEEE 802.3an standard.
- 1000BASE-T and 100BASE-TX as per the IEEE standard 802.3-2012 (Ethernet). Incorporates various IEEE Standards previously published separately. Institute of Electrical and Electronic Engineers (IEEE).
- NBASE-T as per the IEEE P802.3bz/D1.1 Draft Standard for Ethernet Amendment
- IEEE 1149.6 standard for Boundary Scan (MDI pins excluded)
- IEEE standard 802.3ap, draft D3.2.
- IEEE standard 1149.1, 2001 Edition (JTAG). Institute of Electrical and Electronics Engineers (IEEE).
- IEEE standard 802.1Q for VLAN.
- IEEE 1588 International Standard, Precision clock synchronization protocol for networked measurement and control systems, 2004-09.
- IEEE P802.1AE/D5.1, Media Access Control (MAC) Security, January 19, 2006.
- IEEE 802.3az Energy Efficient Ethernet Amendment to IEEE 802.3, October 2010.
- IEEE standard 802.1BR D3.3 - February 20, 2012.
- IEEE 802.Qbg - Amendment 21: Edge Virtual Bridging. 5 July 2012.

PCI-SIG Specifications:

- PCI Express* 2.0 Card Electromechanical Specification
- PCI Express 3.0 Base specification
- PCI Bus Power Management Interface Specification, Rev. 1.2, March 2004.
- PICMG3.1 Ethernet/Fibre Channel Over PICMG 3.0 Draft Specification January 14, 2003 Version D1.0.
- Single Root I/O Virtualization and Sharing, Revision 1.1, September 8, 2009.

IETF Specifications:

- IPv4 specification (RFC 791)
- IPv6 specification (RFC 2460)
- TCP specification (RFC 793)
- UDP specification (RFC 768)
- ARP specification (RFC 826)

IETF Drafts:

- VXLAN — A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks: draft-mahalingam-dutt-dcops-vxlan-03. February 22, 2013
- NVGRE — Network Virtualization using Generic Routing Encapsulation: draft-sridharan-virtualization-nvgre-02. February 24, 2013

Manageability Documents:

- DSP0277 1.0.0 Secured Messages using SPDM Specification, 18 Sep 2020
- DSP0276 1.0.0 Secured Messages using SPDM over MCTP Binding Specification, 16 Nov 2020
- DSP0275 1.0.0 Security Protocol and Data Model (SPDM) over MCTP Binding Specification, 22 Dec 2019
- DSP0274 1.1.1b Security Protocol and Data Model (SPDM) Specification, 25 May 2021
- DSP0267 1.1.0 Platform Level Data Model (PLDM) for Firmware Update Specification, 4 Dec 2019
- DSP0261 1.2.3 NC-SI over MCTP Binding Specification, 25 May 2021
- DSP0249 1.1.0 Platform Level Data Model (PLDM) State Set Specification, 3 Mar 2020
- DSP0248 1.2.1 Platform Level Data Model (PLDM) for Platform Monitoring and Control Specification, 28 Jul 2021
- DSP0245 1.3.0 Platform Level Data Model (PLDM) IDs and Codes Specification, 28 Nov 2018
- DSP0241 1.0.0 Platform Level Data Model (PLDM) Over MCTP Binding Specification, 23 Apr 2009
- DSP0240 1.1.0 PLDM Base Specification, 11 Feb 2021
- DSP0239 1.8.0 Management Component Transport Protocol (MCTP) IDs and Codes, 14 Jun 2021
- DSP0238 1.2.0 MCTP PCIe VDM Transport Binding Specification, 18 Apr 2021
- DSP0237 1.2.0 Management Component Transport Protocol (MCTP) SMBus/I²C Transport Binding Specification, 6 Apr 2020
- DSP0236 1.3.1 Management Component Transport Protocol (MCTP) Base Specification, 4 Sep 2019
- DSP0222 1.1.1 Network Controller Sideband Interface (NC-SI) Specification, 25 May 2021
- DSP0218 1.1.0 PLDM for Redfish Device Enablement, 11 Feb 2021
- System Management Bus (SMBus) Specification, SBS Implementers Forum, Ver. 2.0, August 2000
- UM10204 — I²C-bus specification and user manual Rev. 5, 9 October 2012

Other:

- Advanced Configuration and Power Interface Specification, Rev 2.0b, October 2002
- EUI-64 specification, <http://standards.ieee.org/regauth/oui/tutorials/EUI64.html>.
- Definition for new PAUSE function, Rev. 1.2, 12/26/2006.
- GCM spec — McGrew, D. and J. Viega, "The Galois/Counter Mode of Operation (GCM)", Submission to NIST, January 2004.
<http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/gcm/gcm-spec.pdf>
- FRAMING AND SIGNALING-2 (FC-FS-2) Rev 1.00
- Fibre Channel over Ethernet Draft Presented at the T11 on May 2007
- Per Priority Flow Control (by Cisco Systems) — Definition for new PAUSE function, Rev 1.2, EDCS-472530
- NBASE-T Physical Layer Specification version 1.1

1.7 Architecture and Basic Operation

1.7.1 Transmit (Tx) Data Flow

Tx data flow provides a high-level description of all data/control transformation steps needed for sending Ethernet packets over the wire.

Table 1-8. Tx Data Flow

Step	Description
1	The software device driver creates a descriptor ring and configures one of the E610's transmit queues with the address location, length, head, and tail pointers of the ring (one of 128 available Tx queues).
2	The software device driver is requested by the TCP/IP stack to transmit a packet, it gets the packet data within one or more data buffers.
3	The software device driver initializes the descriptor(s) that point to the data buffer(s) and have additional control parameters that describes the needed hardware functionality. The host places that descriptor in the correct location in the appropriate Tx ring.
4	The software device driver updates the appropriate Queue Tail Pointer (TDT).
5	The E610's DMA senses a change of a specific TDT and as a result sends a PCIe request to fetch the descriptor(s) from host memory.
6	The descriptor(s) content is received in a PCIe read completion and is written to the appropriate location in the descriptor queue.
7	The DMA fetches the next descriptor and processes its content. As a result, the DMA sends PCIe requests to fetch the packet data from system memory.
8	The packet data is received from PCIe completions and passes through the transmit DMA that performs all programmed data manipulations (various CPU offloading tasks as checksum offload, TSO offload, etc.) on the packet data on the fly.
9	While the packet is passing through the DMA, it is stored into the transmit FIFO. After the entire packet is stored in the transmit FIFO, it is then forwarded to transmit switch module.
10	The transmit switch arbitrates between host and management packets and eventually forwards the packet to the security module.
12	The MAC appends the L2 CRC to the packet and delivers the packet to the integrated PHY.
13	The PHY performs the PCS encoding, scrambling, Low-Density Parity Check (LDPC) encoding, and the other manipulations required to deliver the packet over the copper wires at the selected speed.
14	When all the PCIe completions for a given packet are complete, the DMA updates the appropriate descriptor(s).
15	The descriptors are written back to host memory using PCIe posted writes. The head pointer is updated in host memory as well if the E610 is configured to do so.
16	An interrupt is generated to notify the software device driver that the specific packet has been read to the E610 and the driver can then release the buffer(s).

1.7.2 Receive (Rx) Data Flow

Rx data flow provides a high-level description of all data/control transformation steps needed for receiving Ethernet packets.

Table 1-9. Rx Data Flow

Step	Description
1	The software device driver creates a descriptor ring and configures one of the E610's receive queues with the address location, length, head, and tail pointers of the ring (one of 128 available Rx queues).
2	The software device driver initializes descriptor(s) that point to empty data buffer(s). The software device driver places these descriptor(s) in the correct location at the appropriate Rx ring.
3	The software device driver updates the appropriate Queue Tail Pointer (RDT).
4	A packet enters the PHY through the copper wires.
5	The PHY performs the required manipulations on the incoming signal such as LDPC decoding, de-scrambling, PCS decoding, etc.
6	The PHY delivers the packet to the Rx MAC.
9	If the packet matches the pre-programmed criteria of the Rx filtering, it is forwarded to an Rx FIFO.
10	The receive DMA fetches the next descriptor from the appropriate host memory ring to be used for the next received packet.
11	After the entire packet is placed into an Rx FIFO, the receive DMA posts the packet data to the location indicated by the descriptor through the PCIe interface. If the packet size is greater than the buffer size, more descriptor(s) are fetched and their buffers are used for the received packet.
12	When the packet is placed into host memory, the receive DMA updates all the descriptor(s) that were used by the packet data.
13	The receive DMA writes back the descriptor content along with status bits that indicate the packet information including what offloads were done on that packet.
14	The E610 initiates an interrupt to the software device driver to indicate that a new received packet is ready in host memory.
15	The software device driver reads the packet data and sends it to the TCP/IP stack for further processing. The software device driver releases the associated buffer(s) and descriptor(s) once they are no longer in use.

NOTE: *This page intentionally left blank.*

Chapter 2 Pin Interface

2.1 Signal Type Definition

Table 2-1. Signal Definitions

Signal	Definition	DC Specification
In	Standard 3.3/1.8 V I/O buffer, functions as input-only signal.	
Out (O)	Standard 3.3/1.8 V I/O buffer, functions as output-only signal.	
t/s	Tri-state is a 3.3 V bi-directional, tri-state input/output pin.	
o/d	Open drain enables multiple devices to share as a wire-OR.	Section 11.5.1.1
A-In	Analog input signals.	Section 11.5.3 and Section 11.5.4
A-Out	Analog output signals.	Section 11.5.3 and Section 11.5.4
A-Inout	Bi-directional analog signals.	
B	Input BIAS.	
NCSI-in	NC-SI 3.3 V input signal.	Section 11.5.1.2
NCSI-out	NC-SI 3.3 V output signal.	Section 11.5.1.2
In-1p2	1.2 V input-only signal. 3.3 V tolerance.	
In-Only	Standard 3.3/1.8 V buffer input-only signal.	
Out-Only	Standard 3.3/1.8 V buffer output-only signal.	
LVDS-O	Low voltage differential signal - output.	
Pup	Pull up.	
Pdn	Pull down.	

2.2 E610 Die Pin Assignments

2.2.1 PCIe Interface Pins

See AC/DC specifications in [Chapter 11, “Electrical/Mechanical Specification”](#).

Table 2-2. PCIe Interface Pins

Pin Name	MCP 10G Ball Name	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
PET_0_P PET_0_N	PET_0_P PET_0_N	A-Out			PCIe Serial Data Output. A serial differential output pair running at 16 GT/s, 8 GT/s, 5 GT/s, or 2.5 GT/s. This output carries both data and an embedded clock that is recovered along with data at the receiving end.
PET_1_P PET_1_N	PET_1_P PET_1_N	A-Out			
PET_2_P PET_2_N	PET_2_P PET_2_N	A-Out			
PET_3_P PET_3_N	PET_3_P PET_3_N	A-Out			
PER_0_P PER_0_N	PER_0_P PER_0_N	A-In			PCIe Serial Data Input. A serial differential input pair running at 16 GT/s, 8 GT/s, 5 GT/s, or 2.5 GT/s. This input carries both data and an embedded clock that is recovered along with data at the receiving end.
PER_1_P PER_1_N	PER_1_P PER_1_N	A-In			
PER_2_P PER_2_N	PER_2_P PER_2_N	A-In			
PER_3_P PER_3_N	PER_3_P PER_3_N	A-In			
PE_CLK_P PE_CLK_N	PE_CLK_P PE_CLK_N	HCSL-in			PCIe Differential Reference Clock In (a 100 MHz differential clock input): This clock is used as the reference clock for the PCIe Tx/Rx circuitry and by the PCIe core PLL to generate clocks for the PCIe core logic.
PE_RESREF	PE_RESREF	A-In			Reference resistor connection. Connect a 200 Ohm 1% 100-ppm/C precision resistor to ground.
PE_WAKE_N	PE_WAKE_N	O/d		Pup	Wake: Pulled low to indicate that a Power Management Event (PME) is pending and the PCIe link should be restored. Defined in the PCIe specifications. Wake should be failsafe IO.
PE_RST_N	PE_RST_N	In	Pup		Power and Clock Good Indication: Indicates that power and the PCIe reference clock are within specified values. Defined in the PCIe specifications. Also called PCIe Reset.

2.2.2 Ethernet Interface Pins

Table 2-3. Ethernet Interface Pins

Pin Name	MCP 10G Ball Name	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
XTAL_I	XTAL_I	A-In			Positive 25 MHz crystal oscillator input. See Chapter 11 for AC/DC specification.
XTAL_O	XTAL_O	A-Out			Positive 25 MHz crystal oscillator output.
ETH_RESREF	ETH_RESREF	A-In			Reference resistor connection. Connect a 200 Ohm 1% 100-ppm/C precision resistor to ground.
ETH_CLK_P ETH_CLK_N	ETH_CLK_P ETH_CLK_N				156.25 MHz differential clock input for SERDES reference clock. connected from on board XTAL Oscillator.

2.2.3 Serial Flash Memory Interface Pins

See AC/DC specifications in [Chapter 11, “Electrical/Mechanical Specification”](#).

Table 2-4. Serial Flash Memory Interface Pins

Pin Name	MCP 10G Ball Name	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
FLSH_SI	FLSH_SI	Out	Pup		Flash Serial Output: Serial data output that should be connected to the Serial Input (SI) of the SPI serial Flash memory. With the dual and quad SPI commands, the SI Pin becomes I/O0 in conjunction with other pins providing a four-bit wide data path to the flash device. To maintain consistency with the SPI nomenclature, the SI (I/O0) pin is referenced as the SI pin unless specifically addressing the Dual-I/O and Quad-I/O modes in which case it is referenced as I/O0.
FLSH_SO	FLSH_SO	In	Pup		Serial data input that should be connected to the Serial Output (SO) from the SPI serial Flash memory. With the dual and quad SPI commands, the SO Pin becomes I/O1 in conjunction with other pins providing a four-bit wide data path to the flash device. To maintain consistency with the SPI nomenclature, the SO (I/O1) pin is referenced as the SO pin unless specifically addressing the Dual-I/O and Quad-I/O modes in which case it is referenced as I/O1.
FLSH_IO2 FLSH_IO3	FLSH_IO2 FLSH_IO3	t/s Pu			FLSH_IO2 and FLSH_IO3 are only used for quad SPI instructions, providing a four-bit wide data path to the flash device.
FLSH_SCK	FLSH_SCK	Out			Flash serial clock: Flash serial clock operates at 50 MHz.
FLSH_CE_N	FLSH_CE_N	Out	Pup		Flash Chip Select Output.

2.2.4 SMBus Interface Pins

See the AC/DC specifications in [Chapter 11, “Electrical/Mechanical Specification”](#).

Table 2-5. SMBus Interface Pins

Pin Name	MCP 10G Ball Name	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
SMBCLK	SMBCLK	O/d		Pup	SMBus Clock: One clock pulse is generated for each data bit transferred. 3.3 V tolerant when device is not powered.
SMBD	SMBD	O/d		Pup	SMBus Data: Stable during the high period of the clock (unless it is a start or stop condition). 3.3 V tolerant when device is not powered.
SMBALRT_N	SMBALRT_N	O/d		Pup	SMBus Alert: Acts as an interrupt pin of a slave device on the SMBus. 3.3 V tolerant when device is not powered.

Note: If the SMBus is disconnected, use the external pull-up value listed.

2.2.5 NC-SI Interface Pins

See AC specifications in [Chapter 11, “Electrical/Mechanical Specification”](#).

Table 2-6. NC-SI Interface Pins

Pin Name	MCP 10G Ball Name	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
NCSI_CLK_IN	NCSI_CLK_IN	NCSI-In		Pdn	NC-SI Reference Clock Input: Synchronous clock reference for receive, transmit, and control interface. It is a 50 MHz clock ± 100 ppm.
NCSI_TX_EN	NCSI_TX_EN	NCSI-In		Pdn	MC Transmit Enable: Indicates that received data from MC is valid.
NCSI_TXD0 NCSI_TXD1	NCSI_TXD0 NCSI_TXD1	NCSI-In		Pup	MC Transmit Data: Data signals from the MC to the E610.
NCSI_CRS_DV	NCSI_CRS_DV	NCSI-Out		Pdn ¹	Carrier Sense/Receive Data Valid (CRS/DV) to MC: Indicates that the data transmitted from the E610 to MC is valid.
NCSI_RXD0 NCSI_RXD1	NCSI_RXD0 NCSI_RXD1	NCSI-Out		Pup	MC Receive Data: Data signals from the E610 to the MC.
NCSI_ARB_IN	NCSI_ARB_IN	NCSI-In		Pdn ¹	NC-SI Arbitration In
NCSI_ARB_OUT	NCSI_ARB_OUT	NCSI-Out			NC-SI Arbitration Out

1. Should be pulled down if NC-SI interface is disabled.

2.2.6 General Purpose I/O (GPIO) Pins

This section provides the pin assignment for GPIO signals. The E610 has a total of 29 GPIO pins that can be configured as Software Definable Pins (SDPs), LED drivers or dedicated hardware functions for connecting to external PHYs or IEEE 1588 auxiliary devices. The E610 offers the flexibility to configure any of the GPIO pins to different modes and associated with different ports.

The GPIO pins fall into three categories.

- **Direct Timing GPIO:** The E610 have four direct, low latency GPIO pins per port (port 0 use SDPs[19:16] and port 1 use SDPs[23:20]) that can be used for accurate timing applications such as 1588 and generate 1PPS pulse. Those pins can be controlled by SW driver or by FW through direct memory access.
- **GPIO/SDP/LED:** These general purpose software definable pins are driven via the internal I/O expander (I/O Widget). The terms GPIO and SDP are used somewhat interchangeably in this document. Application of these pins is defined by the link topology net list. Refer to the reference design section for supported mapping.
- **Legacy GPIO access:** To support SW driver legacy, some of the SDPs and MDIO / I2C pins can be controlled directly by SW driver through SW MMIO registers. e.g. use to run I2C protocol over SDP pins. LEDs functionality can be activated by the HW automatically.

See [Section 3.4](#) for more details on configurable SDPs.

Table 2-7. GPIO Pins

Pin Name	MCP 10G Ball Name	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
SDP00 SDP01 SDP02 SDP03 SDP04 SDP05 SDP06 SDP07 SDP08 SDP09 SDP10 SDP11 SDP12 SDP13 SDP14 SDP15	SDP00 SDP01 SDP02 SDP03 SDP04 SDP05 SDP06 SDP07 SDP08 SDP09 SDP10 SDP11 SDP12 SDP13 SDP14 SDP15	t/s Pu			General purpose 3.3 V I/Os control by I/O widget. SDPs can be used to connect LEDs, low speed optical module interfaces, external PHY control, interrupts and other general purpose. Refer to the reference design section for supported uses. The first eight SDPs are used as LED by default. It is controlled either by hardware to support legacy products or by firmware through I/O widget. SDP[2:0] are mapped to port0. As per OCP NIC3.0 only 3 LED implemented per port. • SDP[2] - Link/Activity. • SDP[1] - 1 Gbps link. • SDP[0] - Max link speed 10/2.5 Gbps. SDP[6:4] are dedicated for Port1 with similar function as port0. SDP [9] and SDP[8] can be used for syncE clock recovery outputs in SCP SKU. Those clocks are divided and pre-filtered recovered clock output. Should be used to feed a SyncE jitter attenuator PLL on board that fed back the XTAL_I pin in bypass mode to support SyncE. LKV functional usage on board: -SDP 14,15 shall be set for package ID of slot 0 and slot 1 accordingly. LKV PTP functional usage on board: -SDP17 will be used to drive 1PPS output to PHY0 GPIO[2] as a TOD_CLK_IN and as TOD_TRIG_IN to PHY0 GPIO[3] for the init phase. -SDP22 will be used to drive 1PPS output to PHY1 GPIO[2] as a TOD_CLK_IN and as TOD_TRIG_IN to PHY1 GPIO[3] for the init phase. SDP13 should be used as output to select 1PPS source (when set to 1, 1PPS clock is driven from LKV ports)
SDP16 SDP17 SDP18 SDP19 SDP20 SDP21 SDP22 SDP23	SDP16 SDP17 SDP18 SDP19 SDP20 SDP21 SDP22 SDP23	t/s Pu			General purpose 3.3 V I/Os for 1588 synchronized timing applications. Each 4 SDPs per ports used by default for low latency 1588 synchronized timing applications and 1PPS. Those pins can be controlled by SW driver or FW and not through I/O Widget. In addition, it can be used to run I2C over SDP pins.
SDP24 SDP25 SDP26 SDP27 SDP28	SDP24 SDP25 SDP26 SDP27 SDP28	t/s Pu	Pdn Pdn		General purpose 3.3 V I/Os for 1588 synchronized timing applications. SDP[28] is used for UART TXD Data output for EMP FW debug.

2.2.7 Miscellaneous Pins

This section provides the pin assignment for other miscellaneous signals.

Table 2-8. Miscellaneous Pins

Pin Name	MCP 10G Ball Name	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
LAN_PWR_GOOD	LAN_PWR_GOOD	In Pu			LAN Power Good. A 3.3 V input signal. A transition from low-to-high initializes the E610 into operation. If not used (POR_BYPASS = 0b), an internal Power-on-Reset (POR) circuit triggers E610 power-up. If the internal POR circuit is used to trigger device power-up, this signal should be connected to 3.3 V. By default, internal POR should be used.
POR_BYPASS	POR_BYPASS	In Pd			Bypass indication as to whether or not to use the internal POR or the LAN_PWR_GOOD pin. When set to 1b, the E610 disables the internal POR circuit and uses the LAN_PWR_GOOD pin as a POR indication. When set to 0b, the E610 uses both the internal POR circuit and the LAN_PWR_GOOD pin; the device is held in reset as long as the POR is active or LAN_PWR_GOOD is low. By default, this pin should be pulled down to VSS and the internal POR used unless the power supply sequencing timing requirements, as defined in the electrical specification chapter could not be met.
AUX_PWR	AUX_PWR	In Pu			Auxiliary Power Available. When set, indicates that auxiliary power is available and the E610 should support the D3 _{COLD} power state if enabled to do so. This pin is latched at the rising edge of LAN_PWR_GOOD.
MAIN_PWR_OK	MAIN_PWR_OK	In Pu			Main Power OK. Indicates that platform main power is up. Must be connected externally.
PCI_DIS_N	PCI_DIS_N	In Pu			This pin is a strapping pin latched while LAN_PWR_GOOD or PE_RST_N or in-band PCIe reset are asserted. If this pin is not connected or driven high during initialization, then all PCI functions as configured from NVM are enabled. If this pin is asserted/driven low during initialization, then all PCI functions that are allowed to be disabled as configured in NVM are disabled.
DEV_DIS_N	DEV_DIS_N	In Pu			This pin is a strapping option pin latched while LAN_PWR_GOOD or PE_RST_N or In-band PCIe reset are asserted. This pin can be either used as a device disable or for disabling the LAN ports and associated functions based on NVRAM configuration. If this pin is not connected or driven high during initialization, then all the LAN ports and associated functions as configured from NVRAM are enabled for normal operation. If this pin is asserted/driven low during initialization then the LAN ports and associated functions as configured from NVRAM are disabled. Asserting this pin disables the entire device if all the LAN ports are configured to be disabled. When the entire device is disabled, the PCIe link is in L3 state, the PHY is in power down mode, and the output buffers are tri-stated. (see Section 4.5 for details).
SENSE_VDD_P SENSE_VDD_N	SENSE_VDD_P SENSE_VDD_N				Power supply observation pins.

2.2.8 External Ethernet PHY Control - MDIO/I²C Interface Signals

This section provides the pin assignment for other miscellaneous signals.

Table 2-9. External Ethernet PHY Control - MDIO/I²C Interface Signals

Pin Name	MCP 10G Ball Name	Type	Internal Pup/Pdn	External Pup/Pdn	Name and Function
I2C_MDIO_DAT0	I2C_MDIO_DAT0	t/s, o/d Pu			Management Data, when configured as an MDIO interface. Bi-directional signal for serial data transfers between the E610 and the PHY management registers. Note: Tri-state buffer, requires an external pull-up device. I²C Data, when configured as 2-wire management interface. Stable during the high period of the clock (unless it is a start or stop condition). Note: Open drain buffer requires an external pull-up device. Mapping of MDIO/I²C interfaces is controlled by the topology net list stored in NVM. Note: MDIO/I²C can be controlled by SW port 0 through MAC MDIO/I²C (legacy driver) or through FW by using I/O Widget.
I2C_MDIO_CLK0	I2C_MDIO_CLK0	O, o/d Pu			Management Clock, when configured as an MDIO interface. Clock output for accessing the PHY management registers. MDC clock frequency is set to 2.4 MHz (default). I²C Clock, when configured as 2-wire management. One clock pulse is generated for each data bit transferred. Mapping of MDIO/I²C interfaces is controlled by the topology net list stored in NVM. Note: This I/O operates as an open drain buffer, and therefore requires an external pull-up device. Note: MDIO/I²C can be controlled by SW port 0 through MAC MDIO/I²C (legacy driver) or through FW by using I/O Widget.
I2C_MDIO_DAT1	I2C_MDIO_DAT1	T/s, o/d Pu			Same as above. MDIO/I ² C for port 1.
I2C_MDIO_CLK1	I2C_MDIO_CLK1	O, o/d Pu			Same as above. MDIO/I ² C for port 1.

Note: If an I²C interface clock/data pair is disconnected, the pins must be pulled up either by an external pull-up resistor or by enabling the internal pull-up.

2.2.9 Testability and Debug Pins

This section provides the pin assignment for JTAG testability interface signals. Access to the JTAG pins is need for Advanced debug. Refer to the design guide for recommended debug connections.

Table 2-10. Testability and Debug Pins

Pin Name	MCP 10G Ball Name	Type	Name and Function
JTCK	JTCK	In, Pu	JTAG Clock Input. Connect pull up resistor for normal operation.
JTDI	JTDI	In, Pu	JTAG Data Input. Connect pull up resistor for normal operation.
JTDO	JTDO	Out	JTAG Data Output.
JTMS	JTMS	In, Pu	JTAG TMS Input. Connect pull up resistor for normal operation.
JRST_N	JRST_N	In, Pd	JTAG Reset Input. Active low reset for the JTAG port. Connect pull down resistor to VSS for normal operation.
EDM_OUT	EDM_OUT	Out	Edge Die Monitor.
EDM_IN	EDM_IN	In	Edge Die Monitor.

2.2.10 PHY0 (10GBASE-T) Pins

This section provides the pin assignment for 10GBASE-T specific interface signals.

Table 2-11. 10GBASE-T PHY0 Connect Pins

Pin Name	MCP 10G Ball Name	Type	Name and Function
		Out	Interrupt output.
	I2C_MDIO_CLK0	In	MDC is the management data clock reference for the serial management interface. A continuous clock stream is not expected. The maximum frequency supported is 2.4 MHz.
	P0_GPIO_0	In/Out	General Purpose I/O. This pin will be an output at chip reset and be an input after chip reset
	P0_GPIO_1	In/Out	General Purpose I/O. This pin will be an output at chip reset and be an input after chip reset
	P0_GPIO_2	In/Out	General Purpose I/O. This pin will be an output at chip reset and be an input after chip reset
	P0_GPIO_3	In/Out	General Purpose I/O. This pin will be an output at chip reset and be an input after chip reset
	I2C_MDIO_DAT0	In/Out	MDIO is the management data. MDIO transfers management data in and out of the device synchronously to MDC. This pin requires a pull-up resistor in a range from 1.5 KW to 10 KW. Connected to LKV Die on package to I2C_MDIO_DAT0.
	P0_CLK_SEL_0	In	Clock Select 00b = 156.25 MHz Oscillator 01b = 25 MHz XTAL 10b = 25 MHz Oscillator 11b - Reserved
	P0_CLK_SEL_1	In	
	P0_CONFIG_0	In	Hardware configuration pins.
	P0_CONFIG_1	In	Hardware configuration pins.
	P0_CONFIG_2	In	Hardware configuration pins.

Table 2-11. 10GBASE-T PHY0 Connect Pins [continued]

Pin Name	MCP 10G Ball Name	Type	Name and Function
	P0_HSDAC_N	Out	Analog test pins.
	P0_HSDAC_P	Out	Analog test pins.
	P0_MDI_0_N	In/Out	Media Dependent Interface.
	P0_MDI_0_P	In/Out	In 10G/5G/2.5GBASE-T and 1000BASE-T modes in MDI configuration, MDIP/N[0] correspond to BI_DA±. In MDIX configuration, MDIP/N[0] correspond to BI_DB±. In 100BASE-TX and 10BASE-T modes in MDI configuration, MDIP/N[0] are used for the transmit pair. In MDIX configuration, MDIP/N[0] are used for the receive pair. MDIP/N should be tied to ground if not used. The device contains an internal 100Ω resistor between the MDIP/N pins.
	P0_MDI_1_N	In/Out	Same as above.
	P0_MDI_1_P	In/Out	
	P0_MDI_2_N	In/Out	Same as above.
	P0_MDI_2_P	In/Out	
	P0_MDI_3_N	In/Out	Same as above.
	P0_MDI_3_P	In/Out	
	P0_RESETN	In	Hardware reset, Active low. Should be connected to SDP24 on board.
	P0_RCLK	Out	25 MHz Recovered Clock Output.
	P0_IREF	In	Analog reference. should be connected to VSS via 4.99K ohm
	P0_XTAL1	In	XTAL mode: 25 MHz Crystal Input Reference clock input mode: 156.25 or 25 MHz Differential Reference Clock Input - positive
	P0_XTAL2	Out	XTAL mode: 25 MHz Crystal Output Reference clock input mode: 156.25 or 25 MHz Differential Reference Clock Input - negative

2.2.11 PHY1 (10GBASE-T) Pins

This section provides the pin assignment for 10GBASE-T specific interface signals.

Table 2-12. 10GBASE-T PHY1 Connect Pins

Pin Name	MCP 10G Ball Name	Type	Name and Function
		Out	Interrupt output.
	I2C_MDIO_CLK0	In	MDC is the management data clock reference for the serial management interface. A continuous clock stream is not expected. The maximum frequency supported is 2.4 MHz.
	P1_GPIO_0	In/Out	General Purpose I/O. This pin will be an output at chip reset and be an input after chip reset
	P1_GPIO_1	In/Out	General Purpose I/O. This pin will be an output at chip reset and be an input after chip reset
	P1_GPIO_2	In/Out	General Purpose I/O. This pin will be an output at chip reset and be an input after chip reset

Table 2-12. 10GBASE-T PHY1 Connect Pins [continued]

Pin Name	MCP 10G Ball Name	Type	Name and Function
	P1_GPIO_3	In/Out	General Purpose I/O. This pin will be an output at chip reset and be an input after chip reset
	I2C_MDIO_DAT0	In/Out	MDIO is the management data. MDIO transfers management data in and out of the device synchronously to MDC. This pin requires a pull-up resistor in a range from 1.5 KW to 10 KW. Connected to LKV Die on package to I2C_MDIO_DAT0.
	P1_CLK_SEL_0	In	Clock Select 00b = 156.25 MHz Oscillator 01b = 25 MHz XTAL 10b = 25 MHz Oscillator 11b - Reserved
	P1_CLK_SEL_1	In	
	P1_CONFIG_0	In	Hardware configuration pins.
	P1_CONFIG_1	In	Hardware configuration pins.
	P1_CONFIG_2	In	Hardware configuration pins.
	P1_HSDAC_N	Out	Analog test pins.
	P1_HSDAC_P	Out	Analog test pins.
	P1_MDI_0_N	In/Out	Media Dependent Interface. In 10G/5G/2.5GBASE-T and 1000BASE-T modes in MDI configuration, MDIP/N[0] correspond to BI_DA±. In MDIX configuration, MDIP/N[0] correspond to BI_DB±. In 100BASE-TX and 10BASE-Te modes in MDI configuration, MDIP/N[0] are used for the transmit pair. In MDIX configuration, MDIP/N[0] are used for the receive pair. MDIP/N should be tied to ground if not used. The device contains an internal 100Ω resistor between the MDIP/N pins.
	P1_MDI_0_P	In/Out	
	P1_MDI_1_N	In/Out	Same as above.
	P1_MDI_1_P	In/Out	
	P1_MDI_2_N	In/Out	Same as above.
	P1_MDI_2_P	In/Out	
	P1_MDI_3_N	In/Out	Same as above.
	P1_MDI_3_P	In/Out	
	P1_RESETN	In	Hardware reset, Active low. Should be connected to SDP24 on board.
	P1_RCLK	Out	25 MHz Recovered Clock Output.
	P1_IREF	In	Analog reference. should be connected to VSS via 4.99K ohm
	P1_XTAL1	In	XTAL mode: 25 MHz Crystal Input Reference clock input mode: 156.25 or 25 MHz Differential Reference Clock Input - positive
	P1_XTAL2	Out	XTAL mode: 25 MHz Crystal Output Reference clock input mode: 156.25 or 25 MHz Differential Reference Clock Input - negative

2.2.12 Power Supplies

See AC specifications in [Chapter 11, “Electrical/Mechanical Specification”](#).

Table 2-13. Power Supply Pins

Pin Name	MCP 10G Ball Name	Type	Name and Function
VDD_3V3_IO		3.3 V	Digital power supply for 3.3 V I/O.
VDD_1V8_IO		1.8 V	Digital 1.8 V power supply.
VDD_1V8_3V3_IO		1.8 V / 3.3 V	Digital power supply for 3.3 V / 1.8 V I/O. In 10G SKU this bump would be connected to VDD_1V8_IO. SFI is connected to VDD_3V3_IO.
VDDA_1V8_EPHY		1.8 V	Analog high power supply for Ethernet SerDes interfaces.
VDD_0V9_EPHY		0.9 V	Analog power supply for Ethernet SerDes interfaces.
VDDA_1V8_PE		1.8 V	PCIe high analog supply.
VDD_PE		0.75 V	Analog/digital power supply for PCIe SerDes interfaces. (0.75 V)
VDD		0.75 V	Digital core logic power supply.
VDD_XTAL		0.75 V	VDD for XTAL.
VDD_0V9_PLL		0.9 V	PLL digital supply (VDDPOST/VDDREF).
VDDA_1V8_PLL		1.8 V	Analog PLL power supply.
VDDA_1V8_XTAL		1.8 V	Analog PLL power supply.
VDD_1V8_EFUSE		1.8 V	Fuse programming voltage. May be connected to VSS or left floating for normal operation.
VDDA_1V8		1.8 V	PVT analog supply.
VSS_PLL		0 V	
VSS_XTAL		0 V	
VSS		0 V	Digital power supply ground.

2.2.13 PHY0 (10GBASE-T) Power Supplies

Table 2-14. PHY0 Power Supply Pins

Pin Name	MCP 10G Ball Name	Type	Name and Function
	P0_AVDD	0.85 V In	Analog supply
	P0_AVDDC	0.85 V In	Analog supply
	P0_AVDDS	0.85 V In	Analog supply
	P0_AVDDT	2.3 V In	Analog supply
	VSS		
	VSS		
	P0_VDD	0.65 V In	
	P0_VHV	1.8V In	High Voltage Fuse Programming. This pin must be left floating for normal operation.
	P0_VDDO	1.8V In	Digital I/O power.
	P0_DVDD_OUT	1.0 V Out	1.V Digital I/O Power - Regulator Out. This should be connected to VDDO when using an internal 1.8V supply.

2.2.14 PHY1 (10GBASE-T) Power Supplies

Table 2-15. PHY0 Power Supply Pins

Pin Name	MCP 10G Ball Name	Type	Name and Function
	P1_AVDD	0.85 V In	Analog supply
	P1_AVDDC	0.85 V In	Analog supply
	P1_AVDDS	0.85 V In	Analog supply
	P1_AVDDT	2.3 V In	Analog supply
	VSS		
	VSS		
	P1_VDD	0.65 V In	
	P1_VHV	1.8V In	High Voltage Fuse Programming. This pin must be left floating for normal operation.
	P1_VDDO	1.8V In	Digital I/O power.
	P1_DVDD_OUT	1.0 V Out	1.V Digital I/O Power - Regulator Out. This should be connected to VDDO when using an internal 1.8V supply.

2.3 Strapping Options

2.3.1 External Pull-Ups

Table 2-16 lists external pull-up resistors and their functionality.

Table 2-16. External Pull-Up Resistors

Signal Name	External Pull-Up/ Pull-Down	Notes
LAN_PWR_GOOD	Pull-up	
PE_WAKE_N	Pull-up	
FLSH_SCK	Pull-down	
FLSH_SI	Pull-down/Pull-up	
SMBD	Pull-up	
SMBCLK	Pull-up	
SMBALRT_N	Pull-up	
NCSI_CLK_IN	Pull-down	
NCSI_CRS_DV	Pull-down	
NCSI_RXD[1:0]	Pull-up	
NCSI_TX_EN	Pull-down	
NCSI_TXD[1:0]	Pull-up	
JTCK	Pull-up	
JTDI	Pull-up	
JTDO	Pull-up	
JTMS	Pull-up	
JTRST_N	Pull-down	

2.4 Ball Out — Top View Through Package

A1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
A	VSS	VSS	PLMDI_0_N	PLMDI_0_P	VSS	PLMDI_1_N	PLMDI_1_P	VSS	PLMDI_2_N	PLMDI_2_P	VSS	PLMDI_3_N	PLMDI_3_P	VSS	P0_MD_1_0_N	P0_MD_1_0_P	VSS	P0_MD_1_1_N	P0_MD_1_1_P	VSS	P0_MD_1_2_N	P0_MD_1_2_P	VSS	P0_MD_1_3_N	P0_MD_1_3_P	VSS	VSS
B	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
C	PLRCL	REFCL	VSS	P0_RE_SETV	VSS	VSS	VSS	VSS	PLAV_CDC	VSS	PLRCL_K	VSS	P0_RC_LK	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
D	PLXTA_L1	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
E	PLXTA_L2	VSS	VSS	PLIRE_F	VSS	PLVD_001	PLVD_002	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
F	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
G	PLHS_DAC_P	PLI1D_1	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
H	PLHS_DAC_N	PLI1D_0	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
J	P0_HS_DAC_P	VSS	VSS	PLIE_D	PLRES_ETN	PLVD_DO_0U	VSS	VSS	PLAY_CDS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
K	P0_HS_DAC_N	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
L	PLI1D_0	VSS	VSS	PLTES_T	PLDE_BUG	PLCLK_SEL1	PLCLK_SEL0	VSS	PLCO_NFG_2	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
M	PLSPH_O_3	PLSPH_O_1	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
N	PLSPH_O_0	PLSPH_O_2	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
P	P0_SPH_O_0	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
R	P0_SPH_O_1	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
T	P0_SPH_O_2	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
U	P0_SPH_O_3	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
V	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
W	RTAL_1	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
Y	RTAL_0	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
AA	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
AB	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
AC	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
AD	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
AE	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
AF	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
AG	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
AH	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
AJ	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
AK	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
AL	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS

Figure 2-1. E610 Package Layout

Chapter 3 Interconnects

3.1 PCI Express (PCIe)

3.1.1 General Overview

The E610 supports Rev 4.0 of the PCIe base specification.

On top of the capabilities required by the PCIe specifications, The E610 supports optional functionality as listed in this section:

- All PCI functions are native PCIe functions.
- Physical Layer:
 - Support for 2.5 GT/s, 5 GT/s, 8 GT/s and 16 GT/s
 - Interface width of 1, 2 or 4 PCIe lanes 25x25 package
 - Full swing and half swing signaling
 - Lane reversal
- Transaction layer mechanisms:
 - 64-bit and 32-bit memory address spaces
 - Removal of I/O BAR (optional)
 - Relaxed ordering
 - Flow control update timeout mechanism
 - ID-based ordering (IDO)
 - Packet sizes: Maximum packet size: 512, Maximum read request size: 2 KB
 - Function-Level Reset (FLR)
 - TLP Processing Hints (TPH)
- Reliability:
 - Advanced Error Reporting (AER)
 - End-to-End CRC (ECRC) generation and checking
 - Recovery from data poisoning
 - Completion Timeout
- Power management:
 - Active state power management (L1 state)
 - Wake capability
 - Latency Tolerance Reporting (LTR)
- DFT and DFM support for high-volume manufacturing.

- The E610 supports the following Extended Capabilities:
 - Advanced Error Reporting (AER)
 - Device Serial Number (DSN)
 - Alternative RID Interpretation (ARI)
 - Single Root I/O Virtualization (SR-IOV)
 - TPH Requester
 - Access Control Services (ACS)
 - Data Object exchange (DOE)
 - Component Measurement and Authentication (CMA)

3.1.2 Transaction Layer

3.1.2.1 Transaction Types Accepted by the E610

Table 3-1 summarizes the transactions accepted by the device and their attributes.

Table 3-1. Transaction Types Accepted by the Transaction Layer

Transaction Type	FC Type	Tx Layer Reaction	Hardware Should Keep Data from Original Packet
Configuration Read Request	NPH	CPLH + CPLD	Requester ID, TAG, attribute
Configuration Write Request	NPH + NPD	CPLH	Requester ID, TAG, attribute
Memory Read Request	NPH	CPLH + CPLD	Requester ID, TAG, attribute
Memory Write Request	PH + PD	-	-
IO Read Request	NPH	CPLH + CPLD	Requester ID, TAG, attribute
IO Write Request	NPH + NPD	CPLH	Requester ID, TAG, attribute
Read Completions	CPLH + CPLD	-	-
Message	PH + PD ¹	-	-

1. MCTP message contains payload

Flow Control Types Legend:

- CPLD — Completion Data Payload
- CPLH — Completion Headers
- NPD — Non-Posted Request Data Payload
- NPH — Non-Posted Request Headers
- PD — Posted Request Data Payload
- PH — Posted Request Headers

3.1.2.2 Size of Target Accesses

3.1.2.2.1 Memory Accesses

Rules for accesses to the CSR space (both memory BAR and MSI-X BAR):

- Write accesses:
 - Zero-length writes have no internal impact (nothing written, no effect such as clear-by-write). The transaction is treated as a successful operation (no error event).
 - CSR writes are 32-bit or 64-bit only. Larger or partial CSR writes are handled as Unsupported Requests - data is dropped and an error is generated per PCIe rules.
- Read accesses:
 - Partial reads with at least one byte disabled are handled as a full read. Any side effect of the full read (such as clear by read) is also applicable to partial reads. The completion on PCIe adheres to the specification rules regarding the number of bytes reported in the completion.
 - Zero-length reads generate a completion, but the register is not accessed and undefined data is returned.
 - CSR reads are 32-bit or 64-bit only.
 - Some 64-bit reads are handled atomically (i.e. not interleaved with any other requests). This applies mainly to reading counters, where all 64-bit need to be read simultaneously. Such registers are explicitly marked in their description.
 - Larger CSR read requests are handled as Unsupported Requests. The completion includes a UR status and an error is generated per PCIe rules.

Rules for accessing the Flash space in the memory BAR or the Expansion ROM BAR:

- Write accesses:
 - Writes to Flash are silently dropped.
- Read accesses:
 - Reads to Flash are 32-bit wide.
 - Partial reads with at least one byte disabled are handled internally as a full read. That is, any side effect of the full read (such as clear by read) is also applicable to partial reads. The completion on PCIe adheres to the specification rules regarding the number of bytes reported in the completion.
 - Zero-length reads generate a completion, but the Flash is not accessed and undefined data is returned.
 - Larger CSR read requests are handled as Unsupported Requests - the completion includes a UR status and an error is generated per PCIe rules.

3.1.2.2.2 I/O Accesses

Rules for accesses to the I/O BAR:

- Write accesses:
 - Write accesses are 32-bit wide.
 - Zero-length writes have no internal impact (nothing written, no effect such as clear-by-write). The transaction is treated as a successful operation (no error event).
 - Other accesses (partial writes, larger writes) are handled as Unsupported Requests - data is dropped and an error is generated per PCIe rules.
- Read accesses:
 - Reads to the I/O BAR are 32-bit wide.
 - Partial reads with at least one byte disabled are handled internally as a full read. That is, any side effect of the full read (such as clear by read) is also applicable to partial reads. The completion on PCIe obeys the specification rules regarding the number of bytes reported in the completion.
 - Larger CSR read requests are handled as Unsupported Requests - the completion includes a UR status and an error is generated per PCIe rules.

3.1.2.2.3 Messages

MCTP messages may contain payload of up to 64 bytes.

3.1.2.2.4 Support for Dynamic Changes

The E610 captures the Bus Number and Device Number per each Configuration Write Request. However, dynamic change of Bus Number or Device Number is not supported. Rather, the PCIe link should be quiescent prior to such a change, including reception of all completion for previous requests.

3.1.2.3 Transaction Types Initiated by the E610

Table 3-2. Transaction Types Initiated by the Transaction Layer

Transaction Type	Payload Size	FC Type
Configuration Read Request Completion	DWord	CPLH + CPLD
Configuration Write Request Completion	-	CPLH
IO Read Request Completion	DWord	CPLH + CPLD
IO Write Request Completion	-	CPLH
Read Request Completion	DWord/QWord	CPLH + CPLD
Memory Read Request	-	NPH
Memory Write Request	<= MAX_PAYLOAD_SIZE	PH + PD
Message	64 bytes ¹	PH

1. MCTP messages contain payload.

Configuration values:

- Max Payload Size — The value of the *Max Payload Size Supported* field in the Device Capabilities Register is loaded from NVM.
 - Hardware default is 512B.
 - System software then programs the actual value into the *Max Payload Size* field of the Device Control Register.
 - Non-ARI device: If not all functions are programmed with the same value, the max payload size used for all functions is the minimum value programmed among all functions.
 - ARI device: *Max_Payload_Size* is determined solely by the setting in Function 0.
- Max Read Request Size — The E610 supports read requests of up to 2 KB.
 - The actual maximum size of a read request is defined as the minimum {2 KB, *Max Read Request Size* field in the Device Control Register}.

The number of outstanding memory read requests is bounded by the following:

- The total number of outstanding requests is not more than 32 requests. These are shared by all sources for memory reads.

3.1.2.3.1 Data Alignment

Requests must never specify an address/length combination that causes a memory space access to cross a 4 KB boundary. The E610 therefore breaks requests into 4 KB-aligned requests (if needed). This does not place any requirement on software. However, if software allocates a buffer across a 4 KB boundary, hardware issues multiple requests for the buffer. Software should consider aligning buffers to a 4 KB boundary in cases where it improves performance. The maximum size of a read request is defined as the minimum {2 KB bytes, *Max_Read_Request_Size*}

The general rules for packet alignment are as follows. Note that these apply to all the E610 requests (read/write):

- The length of a single request does not exceed the PCIe limit of *MAX_PAYLOAD_SIZE* for write and *MAX_READ_REQ* for read.
- The length of a single request does not exceed the E610 internal limitations.
- A single request does not span across different memory pages as noted by the 4 KB boundary alignment previously mentioned.

If a request can be sent as a single PCIe packet and still meet the general rules for packet alignment, it is not broken at the cache line boundary but rather sent as a single packet. However, if any of the three general rules require that the request is broken into two or more packets, the request is broken at the cache line boundary.

For requests with data payload, if the payload size is larger than (*MAX_PAYLOAD_SIZE* - *CACHELINE_SIZE*), the request is broken into multiple TLPs starting at the first cache line boundary following the (*MAX_PAYLOAD_SIZE* - *CACHELINE_SIZE*) bytes. For example, if *MAX_PAYLOAD_SIZE* = 256 bytes and *CACHELINE_SIZE* = 64 bytes, a 1 KB request starting at address 0x...10 is broken into TLPs such that the first TLP contains 240 bytes of payload (since 240B + 10h = 256B is on cache line boundary).

The system cache line size is controlled by the *PCI_CNFG2.CACHELINE_SIZE* bit, loaded from the NVM. Note that the Cache Line Size Register in the PCI configuration space is not related to the *PCI_CNFG2.CACHELINE_SIZE* and is solely for software use.

3.1.2.4 Messages

3.1.2.4.1 Received Messages

Message packets are special packets that carry a message code. The upstream device transmits special messages to the E610 by using this mechanism. The transaction layer decodes the message code and responds to the message accordingly.

Table 3-3. Supported Messages in the E610 — as a Receiver

Message Code [7:0]	Routing r2r1r0	Message	E610 Later Response
0x14	100b	PM_Active_State_NAK	Accepted
0x19	011b	PME_Turn_Off	Accepted
0x40, 0x41, 0x43, 0x44, 0x45, 0x47, 0x48	100b	Ignored messages (used to be hot-plug messages)	Silently drop
0x50	100b	Slot power limit support (has one DWord data)	Silently drop.
0x7E	000b, 010b, 011b, 100b	Vendor defined type 0	Drop and handle as an Unsupported request.
0x7F	100b	Vendor defined type 1	Silently drop.
0x7F	010b, 011b, 000b	Vendor defined type 1	Send to MCTP reassembly if Vendor ID = 0x1AB4 (DMTF) and VDM code = 0000b (MCTP). Otherwise, silently drop.
0x00	011b	Unlock	Silently drop.

3.1.2.4.2 Transmitted Messages

The transaction layer is also responsible for transmitting specific messages to report internal/external events (such as interrupts and PMEs).

Table 3-4. Supported Messages in the E610 — as a Transmitter

Message Code [7:0]	Routing r2r1r0	Message
0x20	100b	Assert INT A
0x21	100b	Assert INT B
0x22	100b	Assert INT C
0x23	100b	Assert INT D
0x24	100b	De- Assert INT A
0x25	100b	De- Assert INT B
0x26	100b	De- Assert INT C
0x27	100b	De- Assert INT D
0x30	000b	ERR_COR
0x31	000b	ERR_NONFATAL
0x33	000b	ERR_FATAL
0x18	000b	PM_PME

Table 3-4. Supported Messages in the E610 — as a Transmitter [continued]

Message Code [7:0]	Routing r2r1r0	Message
0x1B	101b	PME_TO_Ack
0x10	100b	LTR
0x7F	000b, 010b, 011b	Vendor Defined Messages (VDM)

3.1.2.4.3 Vendor Defined Messages (VDM)

The following vendor defined messages are supported:

- DMTF MCTP

3.1.2.4.3.1 MCTP VDMs

MCTP VDMs are supported as both initiator and target. The following header fields are involved (see Figure 3-5):

- Fmt — Set to 11b to indicate 4 DWord header with data.
- Type
 - [4:3] Set to 10b to indicate a message.
 - [2:0] Routing r2r1r0 = 000b, 010b or 011b.
- Traffic Class — Set to 000b.
- TLP Digest — Set to 0b (no ECRC) or 1 (ECRC) according to the PCI_CAPSUP.ECRC_MCTP_GEN bit.
- Error Present - Set to 0.
- Attributes[1:0] — Set to 01 (no snoop).
- Tag field — Indicates this is an MCTP packet and the size of padding to DWord alignment added.
- Message code = 0x7F (Type 1 VDM).
- Destination ID — captures the target B/D/F for Route by ID. Otherwise, reserved.
- Vendor ID = 0x1AB4 (DMTF).

Table 3-5. MCTP Over PCIe Header Format

FMT 011	Type 10r2r1r0 ¹	R	TC 000	R	A t t r ²	R	T H 2	T D 2	E P 2	Attr [1:0] 2	AT 00	Length 00_000x_xxxx
PCI Requester ID								PCI Tag Field			Message Code	
								R	Pad Len	MCTP VDM code - 0000b	Vendor Defined = 0111_1111b	
PCI Target ID (For Route by ID messages, otherwise = Reserved)								Vendor ID = 0x1AB4 (DMTF)				

1. r2r1r0 =
000b: Route to Root Complex
010b: Route by ID
011b: Broadcast from Root Complex
2. TD = 0, EP = 0, IC = 0, TH = 0, Attr[2:0] = 0 for sent packets and is ignored for received packets.

3.1.2.5 Transaction Attributes

3.1.2.5.1 Traffic Class (TC) and Virtual Channels (VC)

The E610 only supports TC = 0 and VC = 0 (default).

3.1.2.5.2 TLP Processing Hints (TPH)

The E610 supports the TPH capability defined in the PCI Express specification. It does not support Extended TPH requests.

Existence of a TLP Processing Hint (TPH) is indicated on the PCIe link by setting the TH bit in the TLP header. Using the PCIe TLP Steering Tag (ST) and Processing Hints (PH) fields, The E610 can provide hints to the root complex about the destination (socket ID) and about data access patterns (locality in Cache) when executing DMA memory writes or read operations.

The E610 exposes a PCIe TPH capability structure (See [Section 9.2.4.5](#)) with no steering table.

See [Section 7.5](#) for details on the usage of TPH.

3.1.2.6 Ordering Rules

The E610 meets the PCIe ordering rules by following the PCI simple device model:

1. Deadlock Avoidance — The E610 meets the PCIe ordering rules that prevent deadlocks:
 - a. Posted writes overtake stalled read requests. This applies to both target and initiator directions. For example, if initiator read requests are stalled due to lack of credits, initiator posted writes are allowed to proceed. On the target side, it is acceptable to timeout on stalled read requests to allow later posted writes to proceed.
 - b. Target posted writes overtake stalled target configuration writes.
 - c. Completions overtake stalled read requests. This applies to both target and initiator directions. For example, if initiator read requests are stalled due to lack of credits, completions generated by the E610 are allowed to proceed.
2. Descriptor/Data Ordering — The E610 ensures that a Rx descriptor is written back on PCIe only after the data that the descriptor relates to is written to the PCIe link. MSI and MSI-X Ordering Rules. System software might change the MSI or MSI-X tables during run-time. Software expects that interrupt messages issued after the table has been updated are using the updated contents of the tables.
 - a. Since software does not know when the tables are actually updated in the E610, a common scheme is to issue a read request to the MSI or MSI-X table (a PCI configuration read for MSI and a memory read for MSI-X). Software expects that any message issued following the completion of the read request, is using the updated contents of the tables.
 - b. Once an MSI or MSI-X message is issued using the updated contents of the interrupt tables, any consecutive MSI or MSI-X message does not use the contents of the tables prior to the change.
3. The E610 meets the rules relating to independence between target and initiator accesses:
 - a. The acceptance of a target posted request does not depend upon the transmission of any TLP.
 - b. The acceptance of a target non-posted request does not depend upon the transmission of a non-posted request.
 - c. Accepting a completion does not depend upon the transmission of any TLP.

3.1.2.6.1 Relaxed Ordering

The E610 takes advantage of the relaxed ordering rules in PCIe. By setting the relaxed ordering bit in the packet header, the E610 enables the system to optimize performance in the following cases:

1. Relaxed ordering for descriptor and data reads — When the E610 issues a read transaction, its split completion has no ordering relationship with the writes from the CPUs (same direction). It should be allowed to bypass the writes from the CPUs.
2. Relaxed ordering for receiving data writes — When the E610 issues receive data writes, it also enables them to bypass each other in the path to system memory because software does not process this data until their associated descriptor writes are done.
3. The E610 cannot relax ordering for the following requests:
 - Descriptor writes
 - Tx head write back
 - Interrupt messages
 - MCTP messages
 - EMP requests

Relaxed ordering is enabled in the E610 by clearing the CTRL_EXT.RO_DIS bit. Relaxed ordering is further controlled through the *Enable Relaxed Ordering* bit in the PCIe Device Control Register (Section 9.2.3.6.5).

3.1.2.6.2 ID-Based Ordering (IDO)

ID-based ordering was introduced in the PCIe rev. 2.1 specification. When enabled, the E610 sets IDO in all applicable TLPs defined in the PCIe specification.

This capability allows a supporting root complex to relax ordering rules for TLPs sent by different requesters.

IDO is enabled when all of the following conditions are met:

- The PCI_CAPSUP.IDO_EN bit is set (Section 8.2.2.5.4).
- The PCIe IDO Request Enable bit (for requests) or the IDO Completion Enable bit (for completions) in Device Control 2 Register is set (Section 9.2.3.6.11).

3.1.2.7 Flow Control

3.1.2.7.1 Flow Control Rules

The E610 only implements the default Virtual Channel (VC0). A single set of credits is maintained for VC0.

Table 3-6. Flow Control Credits Allocation

Credit Type	Operations	Number of Credits (Dual Port)
Posted Request Header (PH)	Target write Message (one unit)	90 header credit units.
Posted Request Data (PD)	Target Write (Length/16 bytes = one) Message (one unit)	90 data credit units.

Table 3-6. Flow Control Credits Allocation [continued]

Credit Type	Operations	Number of Credits (Dual Port)
Non-Posted Request Header (NPH)	Target read (one unit) Configuration read (one unit) Configuration write (one unit)	8 credit units (to enable concurrent target accesses to both LAN ports).
Non-Posted Request Data (NPD)	Configuration write (one unit)	90 credit units.
Completion Header (CPLH)	Read completion (N/A)	Infinite (accepted immediately).
Completion Data (CPLD)	Read completion (N/A)	Infinite (accepted immediately).

Rules for FC updates:

- UpdateFC packets are sent immediately when a resource becomes available.
- The E610 follows the PCIe recommendations for frequency of UpdateFC FCPs.
- Specific rules apply in L0 or L0s link states. See PCIe specification.

3.1.2.7.2 Flow Control Timeout Mechanism

The E610 implements the optional flow control update timeout mechanism. See the PCIe specification.

3.1.2.8 End-to-End CRC (ECRC)

The E610 supports ECRC as defined in the PCIe specification. The following functionality is provided:

- Inserting ECRC in all transmitted TLPs:
 - The E610 indicates support for inserting ECRC in the *ECRC Generation Capable* bit of the PCIe configuration registers. This bit is loaded from the *ECRC Generation NVM* bit.
 - Inserting ECRC is enabled by the *ECRC Generation Enable* bit of the PCIe configuration registers. VFs follow the behavior of their PF.
 - ECRC is not added to MCTP messages (per MCTP specification) unless the `PCI_CAPSUP.ECRC_MCTP_GEN` bit is set.
- ECRC is checked on all incoming TLPs. A packet received with an ECRC error is dropped. Note that for completions, a completion timeout occurs later (if enabled).
 - The E610 indicates support for ECRC checking in the *ECRC Check Capable* bit of the PCIe configuration registers. This bit is loaded from the *ECRC Check NVM* bit.
 - Checking of ECRC is enabled by the *ECRC Check Enable* bit of the PCIe configuration registers. ECRC checking is done if enabled by at least one physical function (enablement is not done via VFs).
- ECRC errors are reported on all physical functions (PFs) enabled for ECRC checking.
- System software can configure ECRC independently per each physical function.

3.1.3 Link Layer

3.1.3.1 ACK/NAK Scheme

The E610 supports two alternative schemes for ACK/NAK rate:

- NAKs are sent as soon as identified.
- ACKs are sent per Section 3.5.3.1 (Table 3-7, Table 3-8, and Table 3-9) in the PCIe Base Specification.

3.1.3.2 Supported DLLPs

The following DLLPs are supported by the E610 as a receiver:

- ACK
- NAK
- PM_Request_Ack
- InitFC1-P
- InitFC1-NP
- InitFC1-Cpl
- InitFC2-P
- InitFC2-NP
- InitFC2-Cpl
- UpdateFC-P
- UpdateFC-NP
- UpdateFC-Cpl

The following DLLPs are supported by the E610 as a transmitter:

- ACK
- NAK
- PM_Enter_L1
- PM_Enter_L23
- InitFC1-P
- InitFC1-NP
- InitFC1-Cpl
- InitFC2-P
- InitFC2-NP
- InitFC2-Cpl
- UpdateFC-P
- UpdateFC-NP

Note: UpdateFC-Cpl is not sent because of the infinite FC-Cpl allocation.

3.1.3.3 Transmit End Data Bit (EDB) Nullifying — End Bad

A TLP may be signaled as EDB or poisoned if during its transmission from the device, an internal memory error is detected that may corrupt the TLP payload.

3.1.4 Physical Layer

3.1.4.1 Link Speed

The E610 supports [PCIe 2.1](#) and [PCIe 3.0](#).

The following configuration controls link speed:

- **PCIe Supported Link Speeds bit** — Indicates the link speeds supported by the E610.
- **PCIe Current Link Speed bit** — Indicates the negotiated link speed.
- **PCIe Target Link Speed bit** — used to set the target compliance mode speed when software is using the *Enter Compliance* bit to force a link into compliance mode. The default value is loaded from the highest link speed supported defined by the above *Supported Link Speeds*.

The E610 does not initiate a hardware autonomous speed change.

The E610 supports entering compliance mode at the speed indicated in the *Target Link Speed* field in the PCIe Link Control 2 register ([Section 9.2.3.6.13](#)). Compliance mode functionality is controlled via the PCIe Link Control 2 register.

3.1.4.2 Link Width

The E610 supports a maximum link width of x4, x2 or x1. The maximum link width is loaded into the *Max Link Width* field of the PCIe Capability register (*LCAP[11:6]*). Hardware default is the x4 link

During link configuration, the platform and the E610 negotiate on a common link width. The link width must be one of the supported PCIe link widths (x1, x2, x4), such that:

- If Maximum Link Width = x4, the E610 negotiates to either x4, x2 or x1.
- If Maximum Link Width = x2, the E610 negotiates to either x2 or x1
- If Maximum Link Width = x1, the E610 only negotiates to x1

When negotiating for x2, or x1 link, the E610 may negotiate the link to reside starting from physical lane 0 or starting from physical lane 2.

The E610 does not initiate a hardware autonomous link width change.

3.1.4.3 Lane Configurations

The E610 supports lane reversal and degraded modes.

The following general rules determine how the device reacts in different cases of lanes configuration:

- If lane 0 is found valid, the E610 does not initiate lane reversal. The link partner (LP) may initiate lane reversal (to end up with an optimal lane width) and the E610 consents with the lane reversal.
- If lane 0 is found invalid, the E610 initiates lane reversal. Lane reversal succeeds if the link partner supports link reversal.
- If the lanes at both ends of the port (i.e. lanes 0 & 3 for x4, lanes 0 & 1 for x2, lane 0 for x1) are invalid, a link is not established.

Note: Some of the configurations or transitions assume lane reversal is done by the link partner. If the link partner does not support a specific transition, the respective configuration is not provided on that system.

Figure 3-1, Figure 3-2, and Figure 3-3 depict the initial link width configuration and link degradation options. In Figure 3-1 and Figure 3-2, the upper part of the Figure describes link options where the Link Partner (LP) and the E610 (NIC) are aligned. The bottom part of the Figure describes link options where the Link Partner and the E610 are reversed in order.

- Figure 3-1 applies when either the Link partner or the E610 is physically set to x4.
- Figure 3-2 applies when either the Link partner or the E610 is physically set to x2 and both are not physically set x4.
- Figure 3-3 applies when both the Link partner or the E610 is physically set to x1.

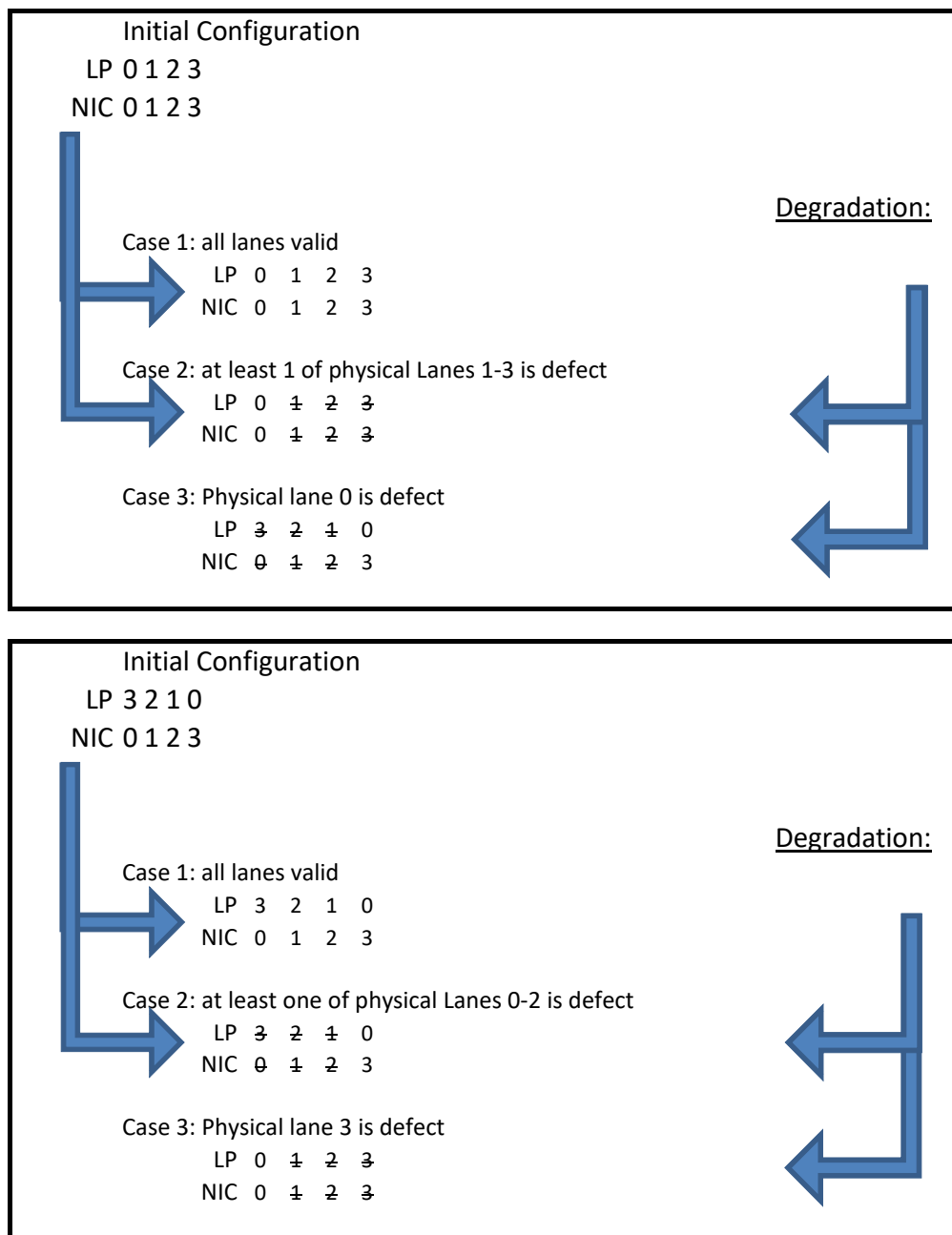


Figure 3-1. Link Width Configurations for a x4 Port

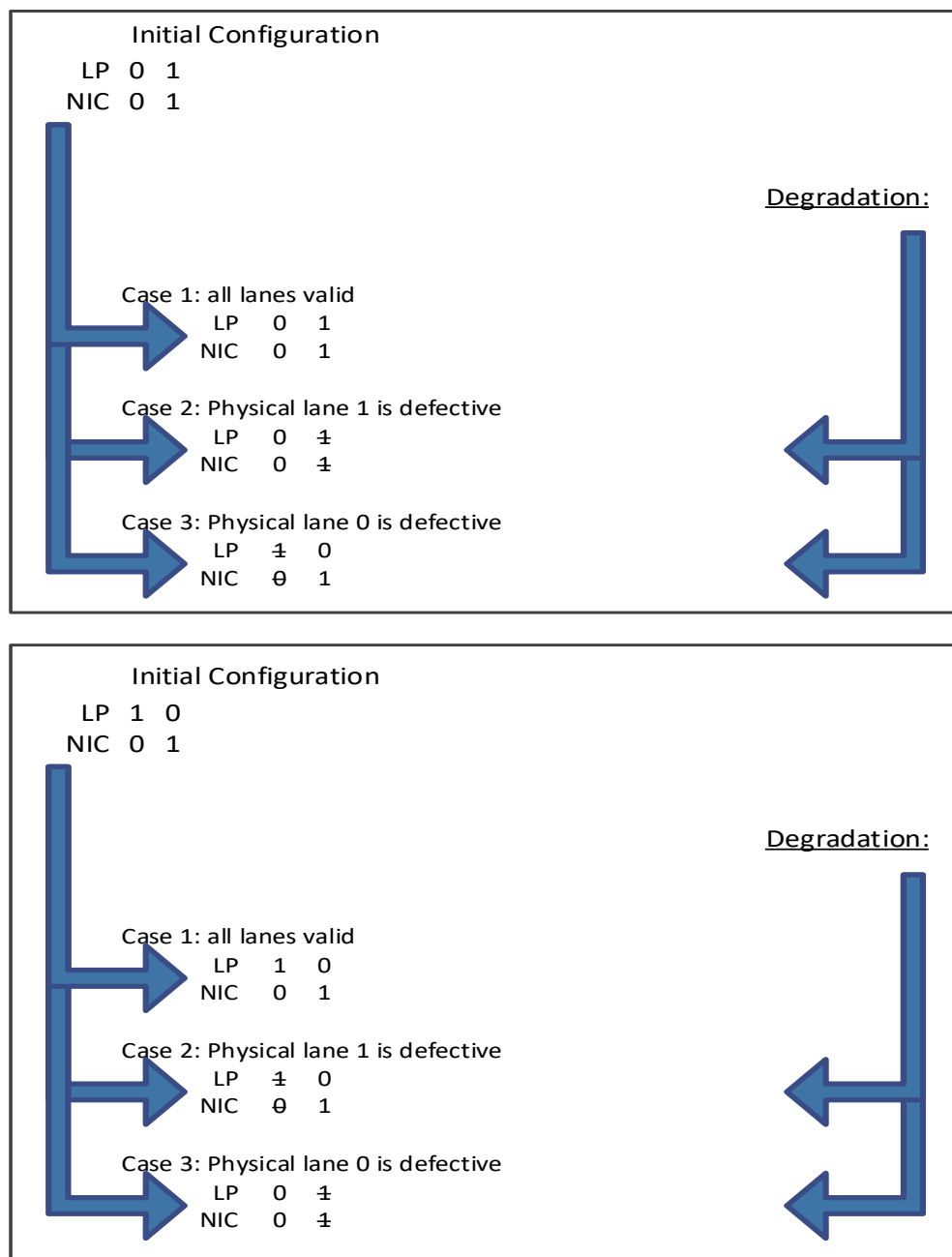


Figure 3-2. Link Width Configurations for a x2 Port

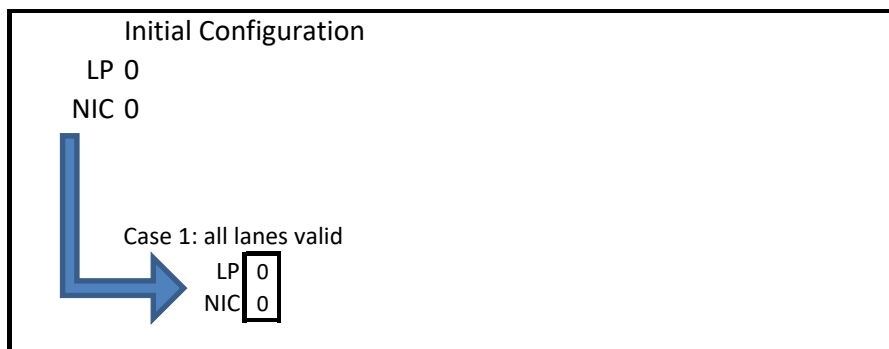


Figure 3-3. Link Width Configurations for a x1 Port

3.1.4.4 Receiver Framing Requirements

This section applies to Gen3 or Gen4 operation only and lists the optional capabilities defined in Section 4.2.2.3.3 (Receiver Framing Requirements) of the PCIe base specification.

The device implements the optional Gen3 receiver framing error checks other than:

- TLP Token length=0
- Mixed order sets across lanes

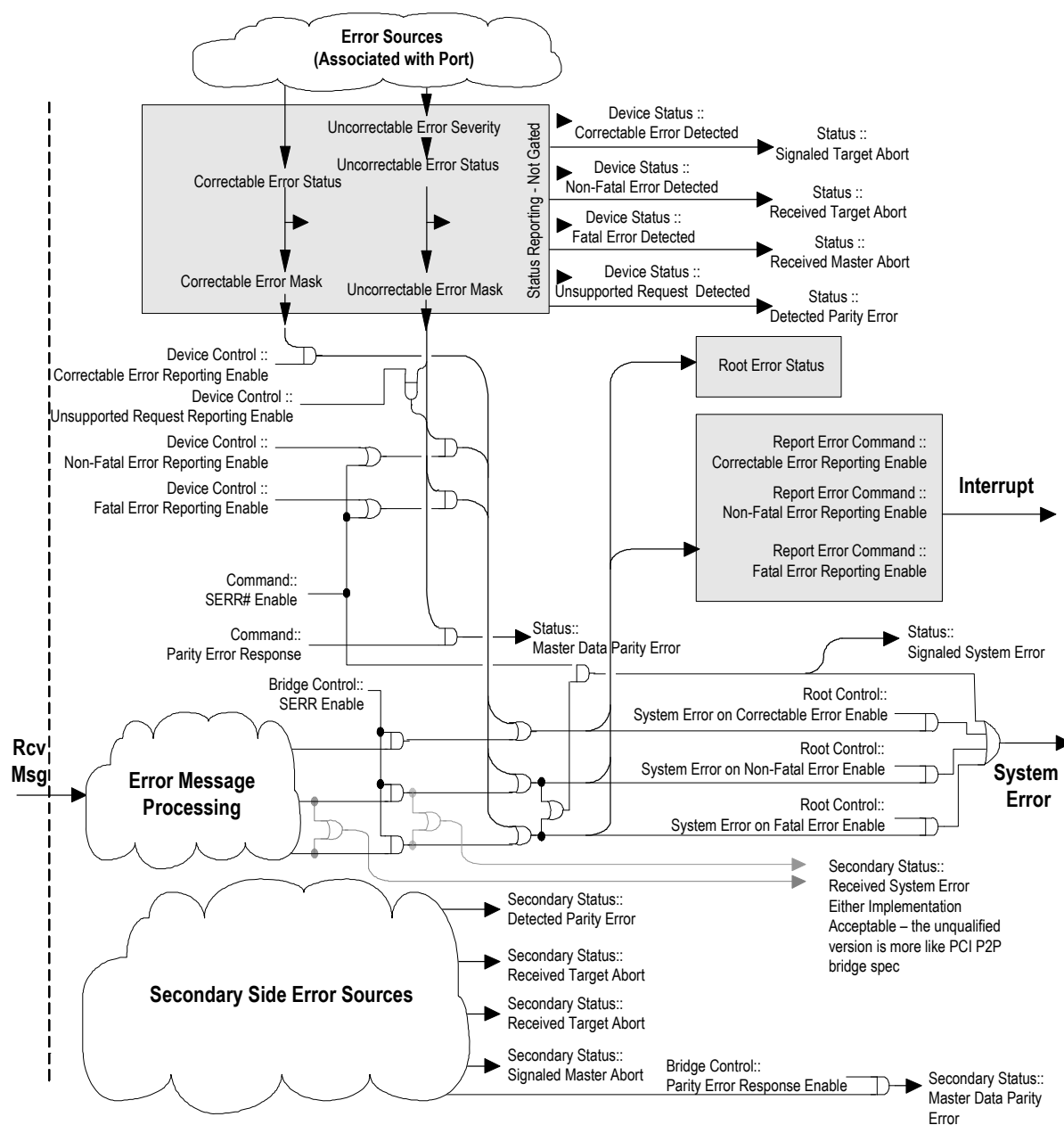
3.1.5 Error Events and Error Reporting

3.1.5.1 General Description

PCIe defines three error reporting paradigms: the baseline capability, the Advanced Error Reporting (AER) capability, and a proprietary mechanism. The baseline error reporting capabilities are required of all PCIe devices and define the minimum error reporting requirements. The AER capability is defined for more robust error reporting and is implemented with a specific PCIe capability structure. Both mechanisms are supported by the E610. The proprietary error reporting mechanism used for error better handled by the software device driver using internal CSRs is described in [Section 3.1.5.8](#).

The *SERR# Enable* and the *Parity Error* bits from the Legacy Command register also take part in the error reporting and logging mechanism.

In a multi-function device, PCIe errors that are not related to any specific function within the device are logged in the corresponding status and logging registers of all functions in that device. [Figure 3-4](#) shows, in detail, the flow of error reporting in the E610.



3.1.5.2 Error Events

Table 3-7 lists the error events identified by the E610 and the response in terms of logging, reporting, and actions taken. Refer to the PCIe specification for the effect on the PCI Status register.

Table 3-7. Response and Reporting of PCIe Error Events

Error Name	Error Events	Default Severity	Action
Physical Layer Errors			
Receiver Error	8b/10b decode errors. - Packet Framing Error.	Correctable Send ERR_CORR	TLP to initiate NAK, drop data. DLLP to drop.
Data Link Errors			
Bad TLP	- Bad CRC. - Illegal EDB. - Wrong sequence number.	Correctable Send ERR_CORR	TLP to initiate NAK, drop data.
Bad DLLP	Bad CRC.	Correctable Send ERR_CORR	DLLP to drop.
Replay Timer Timeout	REPLAY_TIMER expiration.	Correctable Send ERR_CORR	Follow LL rules.
REPLAY NUM Rollover	REPLAY NUM rollover.	Correctable Send ERR_CORR	Follow LL rules.
Data Link Layer Protocol Error	Violations of flow control initialization protocol.	Uncorrectable Send ERR_FATAL	
TLP Errors			
Poisoned TLP Received	TLP with error forwarding.	Uncorrectable ERR_NONFATAL Log Header	If completion TLP: Error is non-fatal (default case): - Send error message if advisory. - Retry the request once and send advisory error message on each failure. - If fails, send uncorrectable error message. Error is defined as fatal: - Send uncorrectable error message.
ECRC Check failed	Failed ECRC check.	Uncorrectable ERR_NONFATAL Log Header	Error is non-fatal (default case): - Send error message if advisory. Error is defined as fatal: - Send uncorrectable error message.
Unsupported Request (UR)	- Receipt of TLP with unsupported request type. - Receipt of an unsupported vendor defined type 0 message. - Invalid message code. - Wrong function number. - Received TLP outside BAR address range. - Receipt of a Request TLP during D3hot, other than Configuration and Message requests. - Received Target Access with illegal data size per Section 3.1.2.2	Uncorrectable ERR_NONFATAL Log header	Send completion with UR. If error is defined as non-fatal (default severity): - Treat as an advisory non-fatal error: Send an ERR_COR Message If error is defined as fatal: - Send ERR_FATAL message

Table 3-7. Response and Reporting of PCIe Error Events [continued]

Error Name	Error Events	Default Severity	Action
Completion Timeout	Completion timeout timer expired.	Uncorrectable ERR_NONFATAL	Error is non-fatal (default case): - Send error message if advisory. Error is defined as fatal: - Send uncorrectable error message.
Unexpected Completion	Received completion without a request for it (Tag, ID, etc.).	Uncorrectable ERR_NONFATAL Log Header	Discard TLP. If error is defined as non-fatal (default severity): - Treat as an advisory non-fatal error: Send an ERR_COR Message If error is defined as fatal: - Send ERR_FATAL message
Receiver Overflow	Received TLP beyond allocated credits.	Uncorrectable ERR_FATAL	Receiver behavior is undefined.
Flow Control Protocol Error	- Minimum initial flow control advertisements. - Flow control update for infinite credit advertisement.	Uncorrectable. ERR_FATAL	Receiver behavior is undefined.
Malformed TLP (MP)	- Data payload exceed <i>Max_Payload_Size</i>. - Received TLP data size does not match length field. - TD field value does not correspond with the observed size. - PM messages that do not use TC0. - Usage of unsupported VC. - Target request crosses a 4KB boundary.	Uncorrectable ERR_FATAL Log Header	Drop the packet, free FC credits.
Completion with Unsuccessful Completion Status		No Action (already done by originator of completion)	Free FC credits.

3.1.5.3 Completion Timeout Mechanism

The E610 supports completion timeout as defined in the PCIe specification.

The E610 controls the following aspects of completion timeout:

- Disabling or enabling completion timeout.
 - The PCIe *Completion Timeout Disable Supported* bit in the Device Capabilities 2 Register ([Section 9.2.3.6.10](#)) is hardwired to 1b to indicate that disabling completion timeout is supported
 - The PCIe *Completion Timeout Disable* bit in Device Control 2 Register controls whether completion timeout is enabled
- A programmable range of timeout values.
 - The E610 supports all four ranges as programmed in the *Completion Timeout Ranges Supported* field of the Device Capabilities 2 Register. The actual completion timeout value is written in the *Completion Timeout Value* field of Device Control 2 Register ([Section 9.2.3.6.10](#)).

The following sequence takes place when completion timeout is detected:

- The appropriate message is sent on PCIe as described in [Table 3-7](#).
- The affected queue or client takes action based on the nature of the original request.
 - If the original request was for Tx packet data, the request and any partial packet completions are dropped.
 - Else, the request is handled same way as malicious requests. An interrupt is issued to the respective PF.
- Software might identify the source of the event (whether due to TLP poisoning, to a completion timeout, or an actual malicious event) by reading the error reporting counters or the performance and statistic counters.

3.1.5.4 Error Forwarding (TLP Poisoning)

If a TLP is received with an error-forwarding trailer, the packet is dropped and is not delivered to its destination. The E610 then reacts as listed in [Table 3-7](#).

The following sequence takes place when a poisoned TLP is received:

- The appropriate message is sent on PCIe as described in [Table 3-7](#).
- An interrupt is issued.
- If the TLP is a completion, a completion timeout follows at some later time. Processing continues as described in [Section 3.1.5.3](#).

System logic is expected to trigger a system-level interrupt to signal the operating system of the problem. Operating systems can then stop the process associated with the transaction, re-allocate memory to a different area instead of the faulty area, etc.

3.1.5.5 Completion with Unsuccessful Completion Status

A completion arriving with unsuccessful completion status (either UR or CA) is dropped and not delivered to its destination. A completion timeout follows at some later time. Processing continues as described in [Section 3.1.5.3](#).

3.1.5.6 Error Pollution

Error pollution can occur if error conditions for a given transaction are not isolated to the error's first occurrence. If the PHY detects and reports a receiver error, to avoid having this error propagate and cause subsequent errors at the upper layers, the same packet is not signaled at the data link or transaction layers. Similarly, when the data link layer detects an error, subsequent errors that occur for the same packet are not signaled at the transaction layer.

3.1.5.7 Blocking on Upper Address

The PCI_UPADD register blocks initiator accesses from being sent out on PCIe if the TLP address exceeds some upper limit. Bits [31:1] correspond to bits [63:33] in the PCIe address space, respectively.

When a bit is set in GLPCI_UPADD[31:1], any transaction in which the corresponding bit in its address is set, is blocked and not sent over PCIe. If all register bits are cleared, there is no effect (in other words, no TLPs are blocked by this mechanism).

The PCI_UPADD register is loaded from NVM with a value allowing all addresses to pass. The software device driver should override this value with a system dependent value.

Processing a blocked transaction:

- Write transaction
 - The transaction is dropped.
 - Set the “Exceeded upper address limit (write requests)” event in the PCIe errors register (see [Section 3.1.5.8](#)).
 - An interrupt is issued as described in [Section 3.1.5.8](#).
- Read transaction
 - The transaction is dropped.
 - Set the “Exceeded upper address limit (read requests)” event in the PCIe errors register (see [Section 3.1.5.8](#)).
 - The originating internal client is notified.
 - The affected queue or client takes action based on the nature of the original request. An interrupt is issued to the respective PF.

3.1.5.8 Proprietary Error Reporting

The PCIe specification defines how to report errors to system software. There are, however, error events that the device driver should be aware of or that the device driver is in better position to handle and recover from. This section describes the mechanism to report PCIe related errors to device drivers.

Several CSRs are dedicated to this functionality, with a separate bit allocated per error type (see [Table 3-8](#)):

- The “PCIe Errors Reported” register (PCI_PCIERR - RO) indicates which errors are reported using this mechanism. It is shared by all PFs. It is loaded from NVM. All non-reserved errors are enabled.
- The “PCIe Interrupt Cause” register (PCI_ICAUSE - RW1C) indicates pending errors for errors set in the PCIe Errors Reported register. It is dedicated per PF.
- The “PCIe Interrupt Enable” register (PCI_IENA - RW) determines if an interrupt should be issued to the respective PCI function on an error event. It is dedicated per PF.

Reporting an error to the PF driver involves the following steps:

- The device checks if the respective bit is set in the PCIe Errors Reported register. If cleared, done. Else, continue
- The respective bit is set in the “PCIe Interrupt Cause” register

- If the respective bit is set in the “PCIe Interrupt Enable” register, an interrupt is issued to the PCI function. The *PCI_EXCEPTION* cause is used.

Table 3-8. PCIe Errors Reported to Device Software

Error Event	Index	Description and Comments	Function Association
Exceeded upper address limit (read requests)	00	See Section 3.1.5.7	Sent to PF
Exceeded upper address limit (write requests)	01	See Section 3.1.5.7	Sent to PF
Reserved	02	Reserved entries	N/A
Poisoned TLP received	03	See Section 3.1.5.4	Sent to PF
Reserved	04-05	Reserved entries	N/A
ECRC error detected	06	ECRC check failed on a received TLP. See Section 3.1.5.7	Sent to all PFs
Unsupported Request - Request Type	07	Request causes an Unsupported Request due to receipt of TLP with unsupported Request Type	Sent to PF
Unsupported Request - Vendor Message	08	Request causes an Unsupported Request due to receipt of an Unsupported Vendor Defined Type 0 Message	Sent to PF unless $r[2:0] = \text{Broadcast from Root Complex}$, in which case sent to all PFs
Unsupported Request - Message Code	09	Request causes an Unsupported Request due to receipt of an invalid Message Code	Sent to PF
Unsupported Request - Function Number	10	Request causes an Unsupported Request due to receipt of a not-supported Function Number	Sent to all PFs
Unsupported Request - Address Range	11	Request causes an Unsupported Request due to receipt of a not-supported Address Range	Sent to PF
Unsupported Request - D3hot	12	Request causes an Unsupported Request due to receipt of a Request TLP during D3hot, other than Configuration and Message requests	Sent to PF
Completer abort - target size	13	Received Target Access with illegal data size per Section 3.1.2.2 (CA)	Sent to PF
Reserved	14-31	Reserved entries	N/A

3.1.6 Performance and Statistics Counters

The E610 incorporates counters to track the behavior and performance of the PCIe interconnect. The E610 implements several types of counters:

- Transaction layer event counters — [Section 3.1.6.1](#)
- Link and Physical layer event counters — [Section 3.1.6.2](#)
- Bandwidth counters — [Section 3.1.6.4](#)
- Latency counters — [Section 3.1.6.5](#)

General characteristics of the counters:

- Software can reset, stop, or start the counters.
- The counters are shared by all PCI functions (“Service” mode of sharing).

Part of the registers that manage the operation of the performance counters are accessed via the PCI_LCBADD and PCI_LCBDATA register pair.

Reading a register via the PCI_LCBADD/PCI_LCBDATA pair is done as follows:

- Write the following values into the PCI_LCBADD register.
 - ADDRESS — The 18-bit register address. See below for the specific address per each register.
 - BLOCK_ID — Defines the sub-unit where the register resides. Use the value 0x7F to access registers mentioned in this section.
 - LOCK — Use if need to gain access in case of multiple agents accessing the PCI_LCBADD/PCI_LCBDATA registers.
- Read the PCI_LCBDATA register.

Note: Although PCI_LCBDATA is a 32-bit register, the registers that maintain the actual count are read as atomic 64-bit reads. The PCI_LCBADD contains the address of the low DW, and reading PCI_LCBDATA returns a 64-bit value.

Writing a register via the PCI_LCBADD/PCI_LCBDATA pair is done as follows:

- Write the following values into the PCI_LCBADD register.
 - ADDRESS — The 18-bit register address. See below for the specific address per each register.
 - BLOCK_ID — Defines the sub-unit where the register resides. For actual values, consult the text below.
 - LOCK — Use if need to gain access in case of multiple agents accessing the PCI_LCBADD/PCI_LCBDATA registers.
- Write to the PCI_LCBDATA register.

3.1.6.1 Event Counters - Transaction Layer

Counters operate in one of the following modes:

- **Count Mode** — The counter increments when the respective event occurred
- **Leaky Bucket Mode** — The counter increments only when the rate of events exceeded a certain value. See [Section 3.1.6.1.2](#) for more details.

The list of events supported by the E610 are listed in [Table 3-9](#).

Table 3-9. PCIe Statistic Events Encoding

Events	Event Mapping (Hex)	Description
Cycles	0x00	Increment on each PCIe clock tick
Transaction Layer Events		
Bad Request TLPs	0x10	Number of bad TLPs arriving to the transaction layer. These include: • Request caused UR • Request caused CA • Malformed TLP
Bad Completions	0x11	Number of bad Completions received. These include: • Unexpected Completion • UR status • CA status
Completion Timeout	0x12	Number of completion timeout events

Table 3-9. PCIe Statistic Events Encoding [continued]

Events	Event Mapping (Hex)	Description
Poisoned TLP	0x13	Number of TLPs received with poisoned data
ECRC Check	0x14	Number of TLPs that foil ECRC check
Link Layer Events		
Retry Buffer Timeout	0x31	Number of replay events that happen due to timeout (does not count replay initiated due to NACK)
Retry Buffer Replay Roll-Over	0x32	Increment when a replay is initiated for more than 3 times
Physical Layer Events		
Receive Error	0x50	Increment when one of the following occurs: 1. Decoder error occurred during training in the PHY. It is reported only when training ends. 2. Decoder error occurred during link-up or till the end of the current packet (in case the link failed). This error is masked when entering/exiting EI.
Surprise Link Down	0x51	Increment when link is unpredictably down (Not because of reset or DFT)

3.1.6.1.1 Count Mode

The following CSR fields control operation of the Count mode:

- Four 32-bit counters `PCI_GSCN_0_3` track events and increment on each occurrence of an event.
 - The four 32-bit counters can also operate in a two 64-bit mode to count long intervals or large payloads.
 - Registers `PCI_GSCN_0_3[0]` and `PCI_GSCN_0_3[1]` form the first 64-bit counter. Registers `PCI_GSCN_0_3[2]` and `PCI_GSCN_0_3[3]` form the second 64-bit counter.
 - The `PCI_GSCL_1.GIO_64_BIT_EN` selects between 32-bit and 64-bit modes.
- The `PCI_GSCL_1.GIO_COUNT_EN_[3:0]` bits enable each of the 4 counters.
 - The enable bits for the two 64-bit counters are `PCI_GSCL_1.GIO_COUNT_EN_0` and `PCI_GSCL_1.GIO_COUNT_EN_2`, respectively.
- The `PCI_GSCL_1.GIO_COUNT_START` bit starts event counting of enabled counters.
- The `PCI_GSCL_1.GIO_COUNT_STOP` bit stops event counting of running counters.
- The `PCI_GSCL_1.GIO_COUNT_RESET` bit resets the event counters.
- The `PCI_GSCL_2` associates an event with each of the 4 counters.
 - In 64-bit mode, the `GIO_EVENT_NUM_[2,0]` fields are used.

3.1.6.1.2 Leaky Bucket Mode

Each of the counters can be configured independently to operate in a leaky bucket mode. When in leaky bucket mode, the following functionality is provided:

- One of four 16-bit Leaky Bucket Counters (LBC) is enabled via the `LBC_ENABLE_[3:0]` bits in the PCIe Statistic Control register #1.
- The LBC is controlled by the `GIO_COUNT_START`, `GIO_COUNT_STOP`, `GIO_COUNT_RESET` bits in the PCIe Statistic Control register #1.

- The LBC increments every time the respective event occurs.
- The LBC is decremented every $T_{\mu s}$ as defined in the *LBC_TIMER_N* field in the PCIe Statistic Control registers #5...#8 (*PCI_GSCL_5_8*).
- When an event occurs and the value of the LBC meets or exceeds the threshold defined in the *LBC_THRESHOLD_N* field in the PCIe Statistic Control registers #5...#8 (*PCI_GSCL_5_8*), the respective statistics counter increments, and the LBC counter is cleared to zero.

3.1.6.2 Link Speed Status

Firmware reads the *PCIE_CAP_LINK_SPEED* field of *LINK_CONTROL_LINK_STATUS_REG* CSR to determine the current PCIe rate:

- Bit 0: 2.5 GT/s
- Bit 1: 5.0 GT/s
- Bit 2: 8.0 GT/s
- Bit 3: 16.0 GT/s

3.1.6.3 Event Counters - Link and Physical Layers

This section describes the performance events for the Link and Physical layers and how to manage the counters associated with these events.

Note: Before using LCB performance counters, the clock gating should be disabled by setting the *PCIE_CLKGATE_DIS* field in the *PCI_GLBL_CNF* register.

The registers responsible for the Link and Physical layers counters are accessed via the *PCI_LCBADD* and *PCI_LCBDATA* register pair.

Two events can be counted concurrently. The event counters include two sets of registers, each managing one event counter. Such pairs are documents as *<register_name>[1:0]*.

The following procedures manage the operation of the event counters (when writing to part of the register, make sure other fields are written with their existing values):

- Resetting the counters configuration:
 - Set the *XPPERFCON.GRST* bit.
 - Clear the *XPPERFCON.GRST* bit (otherwise the logic stays in reset).
- Setting an event:
 - Write 0x0...0 to the *XPPMCL[1:0]* registers
 - Set the *XPPMR[1:0].CENS* field to 0x1
 - Set the *XPPMR[1:0].CNTMD* field to 0x1
 - Set the *XPPMER[1:0].XPRSCA* field to 0x1
 - Set the event according to [Table 3-10](#).
- Starting a count:
 - Set the *XPPERFCON.GCE* bit.

- Stopping a count:
 - Clear the XPPERFCON.GCE bit.
- Reading the count (note: reading the counter clears their values):
 - Read the respective XPPMDH[1:0] and XPPMDL[1:0] register pair in a single 64-bit aligned access.

Table 3-10 defines the Link and Physical Layer events.

Table 3-10. Link and Physical Layer Performance Events

Event	Description	Register Field
Uncorrectable Errors	Counts the total number of Uncorrectable Errors.	XPPMER[1:0].CNTUCERR
Correctable Errors	Counts the total number of Correctable Errors.	XPPMER[1:0].CNTCERR
Tx L0s state utilization	Counts the number of entries to L0s on the Tx lanes.	XPPMER[1:0].TXLOSU
Rx L0s state utilization	Counts the number of entries to L0s on the Rx lanes.	XPPMER[1:0].RXLOSU
Link Utilization	Counts clocks that a port is receiving data. If one counter counts receiver errors and another counter counts Link Utilization, a bit error rate can be calculated.	XPPMER[1:0].LNKUTIL
Recovery State Utilization	Counts the number of entries to Recovery state.	XPPMER[1:0].RECOVERY
ASPM L1 state utilization	Counts the number of entries to ASPM L1 state (i.e. initiated by the device).	XPPMER[1:0].L1
SW L1 state utilization	Counts the number of entries to L1 state initiated by software.	XPPMER[1:0].SWL1
Tx and Rx L0s utilization	Counts number of events where both Tx and Rx are in L0s state.	XPPMER[1:0].RXLOSTXLOSU
NAK DLLP received	Counts number of received NAK DLLPs.	XPPMER[1:0].NAKDLLP

3.1.6.4 Bandwidth Counters

The bandwidth counters measure total payload bytes transferred over the PCIe link. Counting is provided per each traffic type (posted, non-posted, completions) per direction (upstream, downstream).

The mechanisms described above hold for the bandwidth counters with the following differences:

- Setting an event:
 - Set the XPPMR[1:0].CENS field to 0x1.
 - Set the XPPMR[1:0].EGS field to 0x2.
 - Set the XPPMR[1:0].FCCSEL field to the desired traffic type (posted, non-posted, completions, or all).
 - Set the XPPMER[1:0].TXRXSEL field to desired values.
 - Set the XPPMER[1:0].XPRSCA field to 0x1.
 - Set the XPPERFCON.GCE field to 0x1.

Registers fields used exclusively by the bandwidth counters:

- XPPMR[1:0].FCCSEL — Selects the desired traffic type (posted, non-posted, completions, or all).
- XPPMER[1:0].TXRXSEL — Selects between monitoring downstream traffic, upstream traffic, or both.

3.1.6.4.1 Register Map

The register fields that control the Link and Physical Layer events are as follows:

Table 3-11. XP PM Compare Low Bits Register (XPPMCL[1:0]) (0x3288, 0x328C)

Field	Bit(s)	Init.	Description
CMPL	31:0	0xFF...F	PM Compare Low Value Low order bits [31:0] for PM compare register[1:0].

Table 3-12. XP PM Data Low Bits Register (XPPMDL[1:0]) (0x32E8, 0x32F0)

Field	Bit(s)	Init.	Description
CNTL	31:0	0x00...00	PM Data Counter Low Value Low order bits [31:0] for PM data counter[1:0].

Note: XPPMDL must be read together with the respective XPPMDH register as a single 64-bit aligned read. The registers are simultaneously cleared on read.

Table 3-13. XP PM Data High Bits Register (XPPMDH[1:0]) (0x32EC, 0x32F4)

Field	Bit(s)	Init.	Description
CNTH	3:0	0x0	PM Data Counter High Value High order bits [35:32] for PM data counter[1:0].
RSVD	31:4	0x0...0	Reserved.

Note: XPPMDH must be read together with the respective XPPMDL register as a single 64-bit aligned read. The registers are simultaneously cleared on read.

Table 3-14. XP PM Response Control Register (XPPMR[1:0]) (0x3294, 0x3298)

Field	Bit(s)	Init.	Description
RSVD	10:0	0x0	Reserved.
CENS	13:11	00b	Counter Enable Source
CNTMD	15:14	00b	Count Mode
EGS	20:19	00b	Event Group Selection
RSVD	31:21	0x0	Reserved.

Table 3-15. XP PM Resource Events Register (XPPMER[1:0]) (0x32AC, 0x32B0)

Field	Bit(s)	Init.	Description
TXRXSEL	1:0	00b	Tx/Rx Select This field selects the traffic direction to monitor: 1xb = Transmit x1b = Receive (from PCIe bus) 11b = Either Transmit or Receive direction.
FCCSEL	4:2	000b	Flow Control Class Select This field selects which flow class for resource event xx1b = Posted x1xb = Non-Posted 1xxb = Completion Note: Setting to 111b counts posted, non-posted, and completion traffic combined.

Table 3-15. XP PM Resource Events Register (XPPMER[1:0]) (0x32AC, 0x32B0) [continued]

Field	Bit(s)	Init.	Description
RSVD	12:5	0x0	Reserved.
LNKUTIL	16:13	0x0	Link Utilization Set to 0x1 to enable.
XPRSCA	20:17	0x0	XP Resource Assignment 0001b = Set All other values are reserved.
RXL0SU	21	0b	Rx L0s State Utilization Event 0b = Disabled 1b = Enabled
TXL0SU	22	0b	Tx L0s State Utilization Event 0b = Disabled 1b = Enabled
CNTCERR	23	0b	Count Correctable Errors 0b = Disabled 1b = Enabled
CNTUCERR	24	0b	Count Uncorrectable Errors 0b = Disabled 1b = Enabled
RECOVERY	25	0b	Recovery State Utilization 0b = Disabled 1b = Enabled
L1	26	0b	ASPM L1 State Utilization 0b = Disabled 1b = Enabled
SWL1	27	0b	Software L1 State Utilization 0b = Disabled 1b = Enabled
RXL0STXL0SU	28	0b	Tx and Rx L0s Utilization 0b = Disabled 1b = Enabled
NAKDLLP	29	0b	NAK DLLP Received 0b = Disabled 1b = Enabled
RSVD	31:30	0x0	Reserved.

Table 3-16. Performance Monitor Local Control Register (XPPERFCON) (0x32C4)

Field	Bit(s)	Init.	Description
GRST	0	0b	Global Reset
GCE	1	0b	Global Count Enable
RSVD	31:2	0x0	Reserved.

3.1.6.5 Latency Counter

The latency counter measures the min, max, or average read latency.

Note: Completion Timeout events are ignored when the latency counter is enabled.

The latency counters are managed via a set of register fields described below (see also [Table 3-17](#), [Table 3-18](#), and [Table 3-19](#)). Each of the following sources of traffic has its separate set of registers and counters:

- 0x0 — Rx LAN descriptor fetch
- 0x1 — Tx LAN descriptor fetch
- 0x4 — Internal cache load
- 0x5 — Internal management engine read
- 0x6 — Tx LAN packet fetch

The registers are accessed via the PCI_LCBADD and PCI_LCBDATA registers.

The register fields that control the latency counter operation are:

Table 3-17. NPQ Control Register - NPQC (0x00000)

Field	Bit(s)	Init.	Description
Reserved	3:0	0x4	Reserved.
PERFMNTRAVG	7:4	0x1	Performance Monitor Average Rate This field sets the averaging rate for all latency average monitors. See definition of NPQRTDLY1.ARTDLY (Table 3-18). This field divided by 16 is the weight W in an exponential moving average. The possible values are 1, 2, 4 or 8, which correspond to averaging rates of 0.0625, 0.125, 0.25 or 0.5, respectively.
PERFMNTREN	8	0b	Performance Monitor Enable This bit should set to enable latency counters. Clearing this bit clears the latency counters.
RTMNTREN	9	0b	Latency Counting Enable This bit should set to enable latency counters. When set, completion timeout events are ignored.
Reserved	19:10	0x0	Reserved.
cfg_inc_5ms_to	20	0b	Add extra 5 ms of completion timeout (for all the ranges) in HIF.
cfg_inc_30ms_to	21	0b	Add extra 30 ms of completion timeout (for all the ranges) in HIF.
cfg_inc_150ms_to	22	0b	Add extra 150 ms of completion timeout (for all the ranges) in HIF.
cfg_inc_2s_to	23	0b	Add extra 2 s of completion timeout (for all the ranges) in HIF.
cfg_force_high_to	24	0b	Force 64 seconds completion timeout value (for all the ranges) in HIF.
Reserved	31:25	0x0	Reserved.

Table 3-18. NPQ Round-Trip Delay 1 Register - NPQRTDLY1 (0x00030; RO)

Field	Bit(s)	Init.	Description
ARTDLY	15:0	0x0	Average Read Requests Round-trip Delay Captures the average read latency experienced since the last counter reset. Latency is measured from time the read request starts until the time the completion starts to arrive. Average is calculated as exponential moving average. That is, the new average $M(n)$ at sample n equals: $M(n) = (W/16) * \text{new\_sample} + (16-W)/16 * M(n-1)$ where W is defined in the NPQC.PERFMNTRAVG field.
Reserved	31:16	0x0	Reserved.

Table 3-19. NPQ Round-Trip Delay 2 Register - NPQRTDLY2 (0x00034; RO)

Field	Bit(s)	Init.	Description
MINRTDLY	15:0	0x0	Minimal Read Requests Round-Trip Delay Captures the minimal read latency experienced since the last counter reset. Latency is measured from time the read request starts until the time the completion starts to arrive.
MAXRTDLY	31:16	0x0	Maximal Read Requests Round-Trip Delay Captures the maximal read latency experienced since the last counter reset. Latency is measured from time the read request starts until the time the completion starts to arrive.

Latencies are measured in cycle counts, where a cycle duration is per [Table 3-20](#).

Table 3-20. Resolution of the Latency Counters

PCIe Operation Speed	Setting of the PCI_CLKCTL.PCI_CLK_DYN Bit	PCIe Operational Link Width	Cycle Duration (ns)
Gen1 (2.5G)	0b	x	8
Gen2 (5.0G)	0b	x	4
Gen3 (8.0G)	0b	x	2
Gen1 (2.5G)	1b	8 lanes	16
Gen2 (5.0G)	1b	8 lanes	8
Gen1 (2.5G)	1b	1 or 4 lanes	32
Gen2 (5.0G)	1b	1 or 4 lanes	16
Gen3 (8.0G)	1b	1 or 4 lanes	8

3.1.7 Data Object Exchange

3.1.7.1 Overview

Data Object Exchange (DOE) PCIe capability is used to allow system software/firmware to perform data object exchanges with The E610 firmware. This capability enables in-band access to system software/firmware using configuration requests to Component Measurement and Authentication (CMA). CMA in turn builds on SPDM.

This bidirectional communication is done over mailboxes, one for each direction. See the flows described in the following sub sections for further detail.

The E610 implements DOE only on PF0. The mailbox contain two buffers, one for each direction. Each buffer is 1KB to enable passing large security keys.

Operating systems/software access to the mailbox is done in an indirect manner as described in the sub-sections that follow. Firmware is accessing the mailbox directly, as a regular memory region.

The mailbox has a FIFO nature. For example, the operating system/software is pushing by writing a single CSR into the mailbox, and firmware is reading the message from this mailbox.

3.1.7.2 DOE Flows

3.1.7.2.1 DOE Setup

1. When MSI/MSI-X is enabled, firmware chooses one of the vectors, and writes the vector number to the GLPCI_DOE_CFG CSR
2. By default, the DOE Busy bit is set. Firmware clears it when it is ready for CMA communication

3.1.7.2.2 Software-to-Firmware Transfer Flow

1. Software checks that the DOE busy bit is clear to ensure that the DOE instance is ready to receive a DOE request
2. Software writes the entire data object a Dword at a time via the DOE Write Data Mailbox register. Hardware counts the amount of Dwords in GLPCI_DOE_REQ.MSG_NUM_DWS register
3. Software sets the DOE Go bit in the DOE Control register
4. Hardware sets the DOE busy bit in the DOE Status register
5. Firmware is interrupted by hardware, and executes the following:
 - a. Check that GLPCI_DOE_DBG.CFG_ERROR is cleared. Otherwise, the message should be ignored.
 - b. Start reading the data object, GLPCI_DOE_REQ.MSG_NUM_DWS Dwords, starting from GLPCI_DOE_REQ_BUF[0].
 - c. Clears the DOE Busy bit in the DOE Status register when it is ready to get the next data object. In case, a response is needed, this is done after placing the response in the response buffer. If Interrupt is enabled for DOE, this action will set the DOE Interrupt Status bit, and issue an MSI/MSI-X interrupt.

3.1.7.2.3 Firmware-to-Software Transfer Flow

1. Firmware checks that the DOE Data Object Ready is clear.
2. Firmware writes the data object, first Dword, first to GLPCI_DOE_RESP_BUF CSR starting instance 0.
3. When the last Dword is written, firmware sets the Data Object Ready bit in the DOE Status register by setting GLPCI_DOE_RESP.RESP_READY_SET bit.
4. Firmware writes the message size in Dwords to GLPCI_DOE_RESP.MSG_NUM_DWS. This is used by hardware to clear the Data Object Ready bit when software has read the entire message. If Interrupt is enabled for DOE, then action will set the DoE Interrupt Status bit and issue an MSI/MSI-X interrupt.
5. Software waits for an interrupt if applicable, checks/polls the Data Object Ready bit, and if it is set, reads data from the DOE Read Data Mailbox and increments the read pointer of the response buffer by writing to the DOE Read Mailbox (indicating a successful read), a Dword at time until the entire DOE Response is read.
6. Software need to confirm DOE Error bit is cleared before and after read of response.

3.1.7.2.4 Abnormal Case Handling

1. If software sets the DOE Abort bit in the DOE Control register, the mailbox is flushed, and the DOE Error in the DOE Status register is cleared.
2. DOE Error is set in the following cases.
 - a. DOE Go bit in DOE Control register is set while EMP is in reset.
 - b. ECC error on the request or response memories.
 - c. Write request to DoE capability registers was poisoned.
 - d. Software attempts to write the request buffer or the "DoE go" bit while busy is high
 - e. Software writes more than 256B request (request buffer overflow)
 - f. Software attempts to read a response while "Data Object Ready" bit is low
 - g. Firmware set GLPCI_DOE_CTRL.DOE_CFG_ERR_SET bit to indicate an error to software
 - h. Software writes DOE Go bit when request buffer is empty.
3. Hardware does not monitor firmware responsiveness. It is up to software to handle the case in which firmware is not responding to its DOE Go command.

3.2 Management Interfaces

The E610 contains three possible interfaces to an external BMC.

- SMBus
- NC-SI (over RMII)
- MCTP (over PCIe or SMBus)

3.2.1 SMBus

SMBus is an optional interface for pass-through and/or configuration traffic between an external BMC and the E610. The SMBus channel behavior and the commands used to configure or read status from the E610 are described in [Section 10.6](#).

The E610 also enables reporting and controlling the device using the MCTP protocol over SMBus. The MCTP interface is used by the BMC to control the NIC and for pass-through traffic. All network ports are mapped to a single MCTP endpoint on SMBus. For information, refer to [Section 10.6](#).

The SMBus specification defines a maximum frequency of 100 KHz. However, when acting as a slave, the E610 can receive transaction with a clock running at up to 1 MHz. When acting as an initiator, it can toggle the clock at 100 KHz or 400 KHz. The speed used is set by the *SMBus Connection Speed* field in the *SMBus Notification Timeout and Flags* NVM word.

3.2.2 NC-SI (over RMII)

The NC-SI interface in the E610 is a connection to an external MC. It operates as a single interface with an external BMC, where all traffic between the E610 and the BMC flows through the interface.

The NC-SI link supports both pass-through traffic between the BMC and the E610 LAN functions, as well as configuration traffic between the BMC and the E610 internal units as defined in the NC-SI protocol. Refer to [Section 10.6.2](#) for information.

The E610 NC-SI interface meets the NC-SI version 1.0.0 specification as a PHY-side device. The E610 complies with the electrical characteristics defined in the NC-SI specification.

The E610 implements MCTP protocol over NC-SI for command and pass-through traffic. See [Section 10.7](#) for details.

3.3 Non-Volatile Memory (NVM)

3.3.1 General Overview

The following conditions specific to the E610 induced an approach to NVM access:

- LAN traffic can be handled only if the EMP code runs.
- For flexibility reasons, the main EMP code is retrieved from the NVM and not from ROM.
- Unless authenticated, the EMP/PE firmware code presents a potential security threat for the system:
 - Modification/tampering firmware code is presently undetectable to the rest of the system.
 - The E610's various virtualization capabilities (like SR-IOV with VFs and PFs) can reach into virtually any VM or VMM address space.
 - A malicious firmware is capable of sending out the harvested data from the server to unauthorized recipients via DMA reads ... again, undetected. The host CPU is not involved in this process.
 - A malicious firmware is capable of injecting malware into the host, bypassing other platform-level access control.
- The E610 controller core IP is part of several products with specific NVM needs, while the NVM design is common for all of them and consequently it has to meet the most strict requirements.

3.3.1.1 Requirements on NVM Access

The basic requirements from NVM access in the E610 include the following:

1. Guarantee that only Intel provided firmware code (EMP, PHY) is run and device critical data is used by the E610.
 - a. The firmware code and device critical data is cryptographically signed and authenticated prior to execution using a two-key-level authentication scheme. Key Authentication Keys (KAKs) are used to sign Image Signing Keys (ISGs), which in turn are used to sign NVM image content. KAKs are stored as SHA512 hashes in the device fuses. ISGs are a part of the NVM image.
 - b. The Intel LT-CSS is used for signing the ISGs using the KAKs and stores the KAK private keys. The Intel Enterprise Digital Signing Service (EDDS) is used for signing the images themselves and to store the ISGs.
 - c. PKCS #1 v2.2 format with 3072-bit RSA keys and SHA-512 hash is used for signing EMSA-PKCS1-v1.5.
2. Protect NVM image against compromised supply chains and at least partial protection against physical attacks such as NVM content replacement en-route. Full protection against physical attacks is not required at the moment but can be requested in future. It implies security on read approach, where the firmware ROM code is the Root of Trust and the authentication is done at every step of the Chain of Trust, i.e. ROM-Miniload-NVM image.
 - a. Authentication of the NVM bank is performed at major resets but not performed on every NVM access. The resets on which the NVM bank is re-authenticated are: POR and EMPR.
 - b. The hash value of the public key is stored in fuses, while the public key is part of the NVM image.

3. Compromised key revocation is supported. Each KAK has a matching revocation bit while each ISK contains a KAK-signed ID that can be cross-referenced against a set of ISK revocation fuses.
4. Protection against rollback to the NVM version with known security issues is provided
 - a. This is achieved by guaranteeing that the NVM image security revision is not decreased during NVM update.
 - b. A way to override the security revision rollback protection with BMC manageability commands is provided unless prohibited by fuse settings.
5. Initial NVM programming is responsible to program blank fuses encoding the hash of the public verification key if fuses are not configured as part of the packaging process.
6. The ROM firmware supports blank flash programming mode, where the flash can be written by host software, and then subsequently verified on device reset. Blank flash programming mode assumes known good device initialization parameters are built into the ROM.
7. Protect NVM update flow from power failure before completion. This implies the image-update procedure of modules uses a double-bank policy.
8. Meet the boot time requirements described in PCIe spec.
9. Reduce the surface for security attacks and protect as much NVM content as possible taking the "white list" approach. Consequently only a small PFA area with system specific settings surviving NVM update is unprotected by design. All other NVM settings are read only and/or signed.
10. Provide a simple way to update NVM shifting responsibility for system specific configuration preservation to the device firmware. This is required to support the update over PLDM DMTF emerging standard and any to any NVM update.
11. Prevent a malicious software from causing permanent damage to the system, and to the NVM or to the E610 in particular so that flash parts or NICs are not returned to Intel. It implies some countermeasure be taken against a malicious software that would excessively write access the flash to precipitate its wear-out.

3.3.1.2 Operational Limitations

The NVM protection method selected in E610 relies on authenticating key sections on initial load, also known as authenticate on read. Protected modules are authenticated before their first usage following a power on reset, in addition to validating image updates prior to committing an update. NVM protection is implemented using a ROM-based firmware digital signature authentication algorithm which uses fuse-encoded hashes of the public verification keys. Authenticating images on every initial load greatly reduces the possibility of unwanted modification to device firmware being introduced from compromised supply chains or physical tampering.

There is small set of PCIe related time critical CSR, which are autoloading speculatively without authentication. This content is verified later at initialization time. If the verification fails, the NVM content is fixed and the error is reported to the software (using GL_MNG_FWSW CSR), with the request to initiate a corresponding reset.

In normal operating mode, NVM write accesses are access controlled by the EMP firmware and cannot be performed directly from the host to the NVM device via the memory mapped accesses. Memory mapped NVM access remains available for NVM read accesses only. For simplicity and flexibility reasons, NVM write accesses (except for VPD) can be initiated only via an admin command or following a BMC command, which are both handled by the EMP.

The ROM-based EMP firmware supports a blank flash programming mode in the event the initial EMP firmware loader detects either a blank, corrupt or invalid NVM image. In blank flash programming mode, host software program can program an image directly to the flash device without any security

limitations, nor is the EMP firmware involved. The host software has direct access to the SPI flash controller with the same permission as the EMP firmware in the blank flash programming mode. It should be noted that the new image will not be functional without a reset, and upon a reset, the image will be authenticated.

When the host debug mode is entered via a dedicated JTAG code (0x3E), the host can remove address protection of the PF space and write into the register that controls the blank Flash programming mode.

Note: This is not an error flow, Thus the main firmware is loaded, the autoload process still occurs and the state of the device may not be the hardware default state.

3.3.2 External Flash

In the E610, the LAN controller core is the main client of the internal SPI controller. The E610's full flash image must also contain a descriptor for the SPI controller as described below.

The SPI controller is responsible for the following tasks:

- Perform arbitration between the different clients.
- Manage the access control to the different regions.
- Reflect a zero-base address to each of the clients such as each client's first address is address zero.
- Enable access for a software tool for global blank Flash programming.

Note: **External Flash must have Descriptor for SPI controller preprogrammed.** The External Flash used by the Device must have Descriptor for SPI controller pre-programmed in the Flash for the device to function. Until the Flash contains this information software tools cannot program a NVM image.

3.3.2.1 E610 NVM Regions

In the E610, the NVM uses a 16 or 32 MB flash and contains the following regions:

- **Descriptor region** — The first 4KB of the flash that contains the following data:
 - Signature
 - Content section (3 DWords)
 - Component section
 - Region section
 - Clients section
 - Soft-Straps

Presence of the descriptor region is a requirement of the SPI Controller block. This region must be pre-programmed before the host or firmware can access the flash device via the SPI controller. This programming may be done off-line during the manufacturing process, or via the host interface using the "bit bang" mode.

- **Core region** — The core NVM used by the IP as described in [Section 6.3](#). It is mapped as Region 3 and has a size of 8-10 MB.

3.3.3 Shadow RAM

NVM modules that meet one of these criteria must also be mirrored internally into a shadow RAM that is loaded once after POR:

1. Software must be able to partially update the module without being forced to rewrite the entire module (like SMBus address, VPD, etc.).

Unlike an EEPROM, Flash devices require rewriting an entire sector even if it comes to updating a single byte. The partial update is first performed against the shadow RAM. Later on, the E610 commits the entire updated shadow RAM into the Flash.

2. On device-level resets (in contrast to function-level resets), the module (or parts of it) is auto-loaded by the E610 into registers that are mapped to the host memory BAR.

Autoload done after PCIe fundamental resets (POR and PERST#) must be completed within a bounded time, and cannot wait for the delays involved by a sector erase operation (hundreds of ms) that could have been initiated just before. Flash read accesses are suspended until a Flash sector erase operation completes. NVM auto-load performed further to device-level resets are done from the internal shadow RAM into the registers, without involving Flash read cycles.

The E610 maintains the first 32 x 4 KB sectors of the Flash area allocated to it for the configuration content that must be mirrored into the shadow RAM. These sectors are organized in two equally sized banks, each one capable of containing the PFA of the shadow RAM contents (See [Figure 3-5](#) – this includes also the init module). These banks are referred to as the basic NVM banks. At any time one of these two banks must be valid or else the E610 is set by hardware default and enters into blank Flash programming mode (refer to [Section 3.3.4.2](#)).

Following a Power On Reset (POR), E610 firmware copies the valid Persistent Shadow RAM bank of the Flash device and Non-persistent Shadow RAM area of the valid NVM Bank into the internal shadow RAM (see figures below), which is made resilient to device-level resets that might occur later on. The valid bank is the lowest indexed bank with the validity field content read as 01b. The NVM Validity field is located at NVM Control Word 1. At any time, the valid bank is referred to as the current basic bank, while the other is referred as the next basic bank. Any further accesses of software to this section of the NVM are directed to the internal shadow RAM. Modifications made to the Persistent shadow RAM area content are then copied by the E610 into the next bank of the NVM, flipping circularly the valid Persistent Shadow RAM bank between bank 0 and bank 1 of the Flash.

This mechanism also provides a way for software to protect an image-update procedure from power down events by establishing a dual-bank policy even when performing a module partial update.

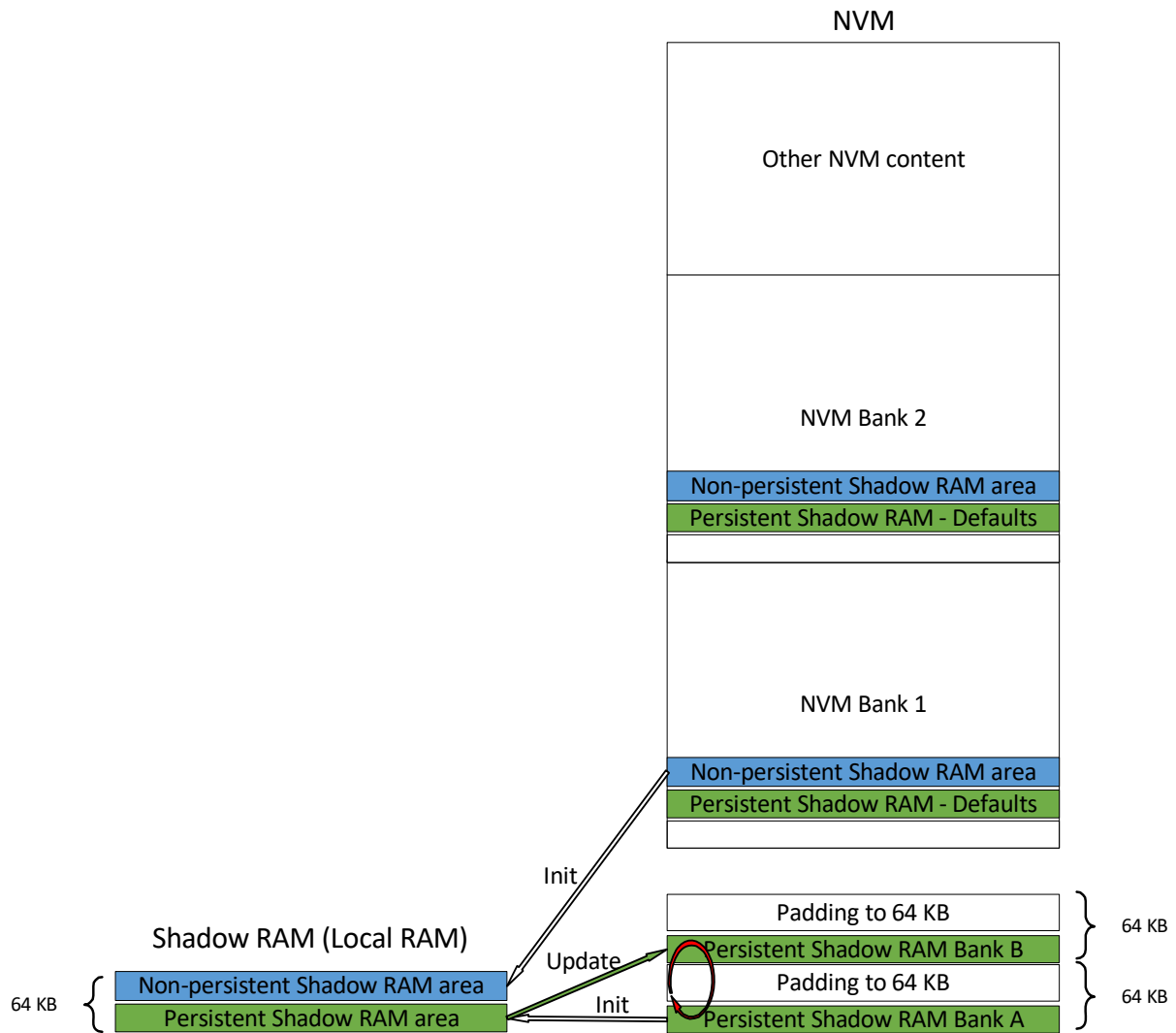


Figure 3-5. NVM Shadow RAM

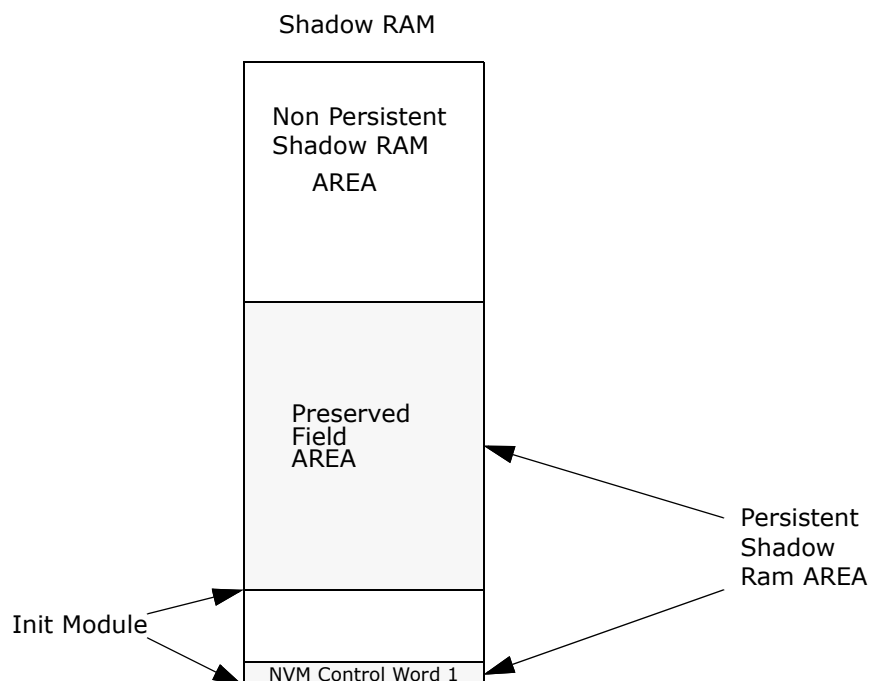


Figure 3-6. Shadow RAM Structure

3.3.4 NVM Access Modes

The NVM is connected to the device through SPI. Every access method uses eventually the SPI.

The NVM is accessed in multiple ways according to the entity and the state of the NVM image. The bullets below list the different access modes.

- Normal access mode
 - Through admin commands
 - VPD register set
 - Memory map
- Blank flash programming mode

3.3.4.1 Normal Mode

For BIOS read accesses and VPD accesses, any read or write access to the NVM by the host must be preceded by taking ownership of the NVM resource via the Request Resource Ownership admin command. Refer to [Section 10.11.2.7.3](#). This prevents the following situations:

- Reading a module that is currently being modified by another entity.
- Concurrent modifying a module contents.

3.3.4.1.1 Normal Read Access

Memory mapped Read accesses to the NVM do not require the EMP being involved. EMP is involved when the read is performed via the NVM Read admin command. Available read accesses are as follows:

- An NVM Read admin command from the PF.
- VPD register set.
- Memory-mapped read via the memory/option ROM BAR. To save host memory addresses, memory BAR access to the NVM is not always available. It is enabled/disabled by setting the *Flash_Expose* bit in the NVM (or setting the `PCI_LBARCTRL.FLASH_EXPOSE` CSR bit).

3.3.4.1.2 Normal Write Access

Write accesses to the NVM are controlled by the EMP. Two accesses are provided:

- An NVM Update admin command from the PF.
- VPD register set. The EMP asserts the Done bit.

NVM write access attempts performed via the memory/option ROM BARs are not performed by the E610, although PCIe transactions are (apparently) normally completed.

3.3.4.2 Blank Flash Programming Mode

The E610 enters local blank Flash programming mode based on the following:

- When a blank Flash is detected. It means that the NVM Validity field (NVM Control Word 1) read from the two basic banks is not equal to 01b.
- When the EMP image read at initialization time does not belong to the E610.
- When the ROM code fails to authenticate the mini-loader image.
- When the mini-loader fails to authenticate the whole NVM bank.
- When the `GLNVM_FLA.LOCKED` bit is cleared. This bit can be cleared from the PF space if host debug mode is entered. It is recommended that Flash programming platforms at manufacturing sites be provided with the JTAG as a back-up means.

This mode is not secure and must be used only at manufacturing time and/or as a last resort to recover from initial mis-configurations. Only host access to the Flash and shadow RAM is guaranteed when in this mode.

It is not recommended to enter this mode at run time because no resource ownership taking is required prior to accessing the NVM. Also, taking resource ownership requires an operational EMP, which is not the case when in this mode.

When the E610 is in this mode (see conditions from a. to e. previously described), the EMP/PE codes are not loaded from the NVM and the EMP/PE remains disabled. The E610 **is not able to exchange any kind of traffic over the lines and no admin command can be posted**. The E610 is in an unknown operational state where only the Flash programming flow is operational.

3.3.5 NVM Update Flows

The flows described in this section only affects normal programming mode. When in the Blank Flash Programming mode, refer to the NVM access procedures described in [Section 3.3.8](#).

It is firmware's responsibility to keep track of Shadow RAM consistency between Internal RAM and Flash.

It is assumed that an NVM update sequence does not last beyond the NVM ownership timeout for a write of three minutes. Refer to [Table 10-105](#). Failing to complete an update (by setting LAST bit in the command flag) within that time, will result in EMPR which will cause a roll-back of the non-completed update.

The firmware is responsible to re-compute and update the software checksum (PFA section with TLV type = 0x3F) each time the shadow RAM content is changed.

3.3.5.1 Flash High-Level Map

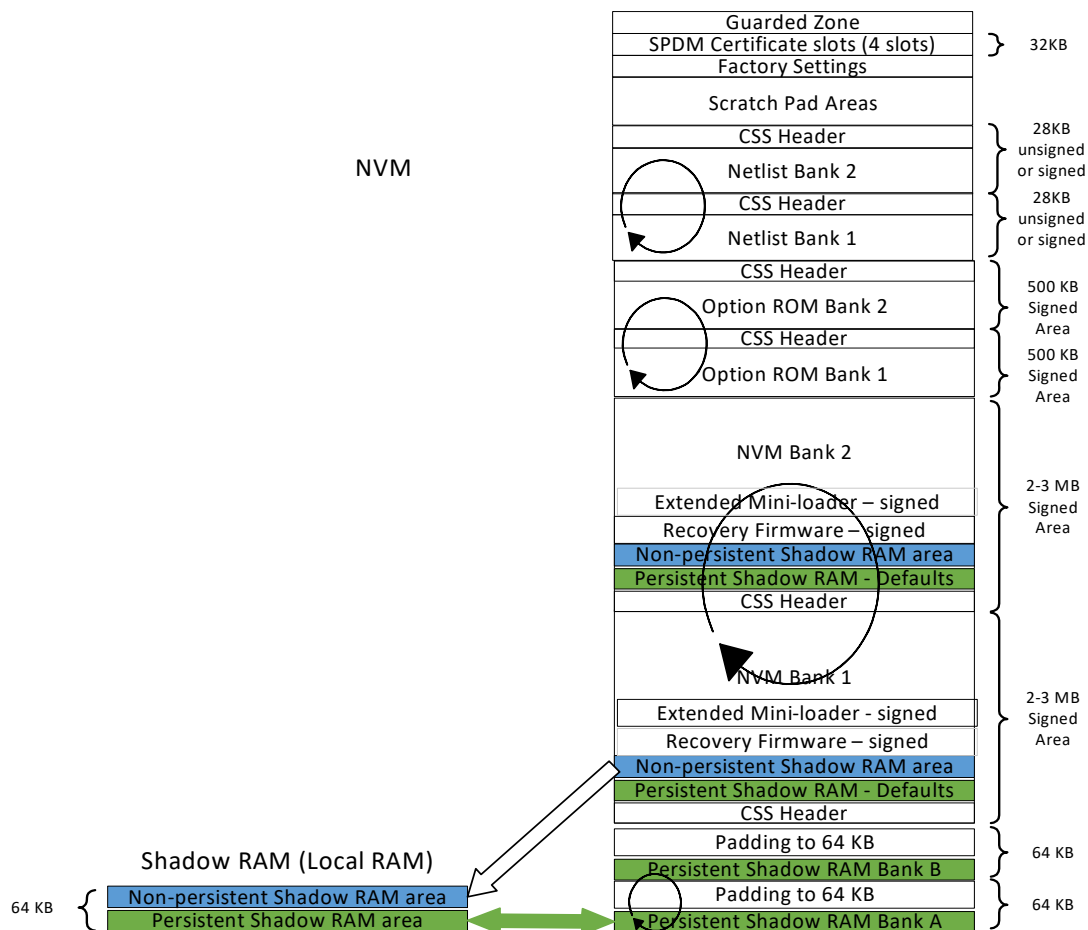


Figure 3-7. Flash High-Level Map

Figure 3-7 shows the high level mapping of the NVM in the E610, while the exact mapping is given in Chapter 6. It is made of the following areas:

- **Two Basic Persistent Shadow RAM Banks A,B** — It contains a modifiable content including NVM Control Word 1 (offset 0x00) along with the Preserved Fields Area (PFA) that includes the VPD area and other modules to be preserved upon a new image release. It may also contain other content for convenience, which is overwritten with the clean copy of the shadow RAM at POR. The current valid bank is mirrored into the internal shadow RAM at POR events. Changes that are made in the shadow RAM are finally dumped into the next bank, which becomes the current valid, and so forth cyclically. Refer to [Section 3.3.5.3](#) and [Section 3.3.5.4](#), and [Section 3.3.5.6](#) for the shadow RAM update flows.
- **Two authenticated NVM banks** — They contain the basic NVM image from which specific images are created. One of the banks is valid, and the other is used for updating. Each area contains:
 - A clean copy of the shadow RAM area, with default values in the Preserved Field Area.
 - The mini-loader image signed along with the time critical configurations.
 - EMP Image area - contains the code to be run by EMP embedded processor.
 - PHY Configuration scripts.
 - EMP Global and Setting modules - these contains basic configurations common to all images.
 - Adaptive NVM metadata - a data structure containing the meta data required to modify an image to match a given configuration.
 - PHY Analog modules (internal Ethernet PHY microcode and configuration). These modules are loaded by the E610 at power-up events only.
 - Recovery mode firmware image, separately signed. See [Section 14.3](#).

See [Section 3.3.5.6](#) for the update flow of the signed area.

- **Two Expansion/Option ROM Areas (OROM area)** — It contains pre-boot code and settings read by BIOS. Refer to [Section 3.3.5.5](#) for the update flow. Pre-boot code is authenticated by BIOS at initialization time before being executed. Pre-boot code is also internally authenticated on update. One of the banks is valid, and the other is used for updating.
- **Netlist Module** — A pair of RW modules used to store the netlist structure.
- **Scratch Pads** — A set of RW areas that are not double banked, where software and firmware can store data for logging and debug.
- **SPDM Certificate Slots** — Contain the device SPDM certificates.

Refer to [Section 6.3](#) for the detailed NVM map.

3.3.5.2 Generic Flows

This section describes the flows that are used as building blocks by other flows.

3.3.5.2.1 Shadow RAM Dump

1. The firmware sets internal bit indicating flash and shadow RAM are inconsistent.
2. The firmware erases the next bank — It erases the contents of the next basic bank sectors.
3. The firmware copies first part of the shadow RAM (From the first word until the end of the PFA) into the next bank sector with the exception of the *Validity* field, which is left as all ones.
4. The firmware checks the Flash write — The new bank content is read and checked to be identical to the shadow RAM contents. This can be done in the course of writing to the Flash using previous step.
 - a. If not identical (such as Flash defect), exit the flow. Refer to [Section 10.11.2.7.5](#).
 - b. If the check was successful (identical), the EMP validates the new bank and invalidates the old bank.
 - The *Validity* field of the new bank is set to 01b. The EMP checks that the *Validity* field is read as written in the Flash. If not, then go to the previous sub-step.
 - The EMP toggles the state of the BANK1VAL bit in the GLNVM_GENS register to indicate that the non-valid bank became the valid one and vice versa.
 - The current (old) bank is invalidated by setting its *Validity* field to 00b.
 - Firmware clears internal bit indicating flash and shadow RAM are inconsistent.

3.3.5.3 VPD Update

3.3.5.3.1 First VPD Area Programming

The VPD capability is exposed on the PCIe interface only if the `PCI_CAPCTRL.VPD_EN` bit is set to 1b, regardless to any other sanity check that is performed on the VPD area contents.

The VPD area and VPD pointer must be written on a blank Flash and must contain a valid contents from this first programming. If VPD tags were modified, it is required to issue a PCIe reset before write accessing the VPD area from PCIe configuration space.

3.3.5.3.2 VPD Area Update from PCIe Configuration Space

The flow described on this section is used once the VPD area contents and pointer have been already programmed in the NVM and loaded into the E610.

1. A PF VPD software performs a VPD write — It sets write offset/data into VPD register set of the relevant PF configuration space, setting the VPD Flag (bit 15 in VPD Address register 0x0E2).
2. Hardware notifies the EMP — It issues an internal VPD access interrupt to the EMP to notify it of the VPD access and of the PF affected.
3. The EMP checks that the VPD write is allowed — It checks that the write offset points to the VPDRW area (and not to a VPD RO area).
 - a. If not, the EMP clears the VPD Flag in the PF configuration space to notify PF VPD software that the transaction completed and then exits the flow.
4. EMP writes the change into shadow RAM.
5. The EMP completes the VPD access to software — The EMP clears the VPD Flag in the PF configuration space to notify PF VPD software that the access completed.
6. The EMP dumps the shadow RAM to flash.
 - The EMP takes ownership over the NVM resources for a write.
 - EMP performs the dump as described in [Section 3.3.5.2.1](#).
 - If the Flash is busy by a previous sector erase operation, flash erase is blocked by the wear out protection mechanism (lack of credits) or if NVM ownership is held by software, it might that the flow needs to be restarted from step 1. at this stage by successive VPD write accesses initiated by VPD software.
 - The EMP releases the NVM ownership.

Notes:

- Users must be made aware that dumping the VPD change into the Flash might take a few hundreds of a ms after the VPD transaction completes to software (by clearing the VPD Flag). As a result, they must wait few seconds before they can shut down the system.
- If the VPD write access is attempted by the host when the E610 has just started a shadow RAM dump ([Step 6](#)), then it might be that the write request times out.

3.3.5.4 Updating Items in the Preserved Fields Area (PFA)

Figure 3-8 describes the PFA update flow.

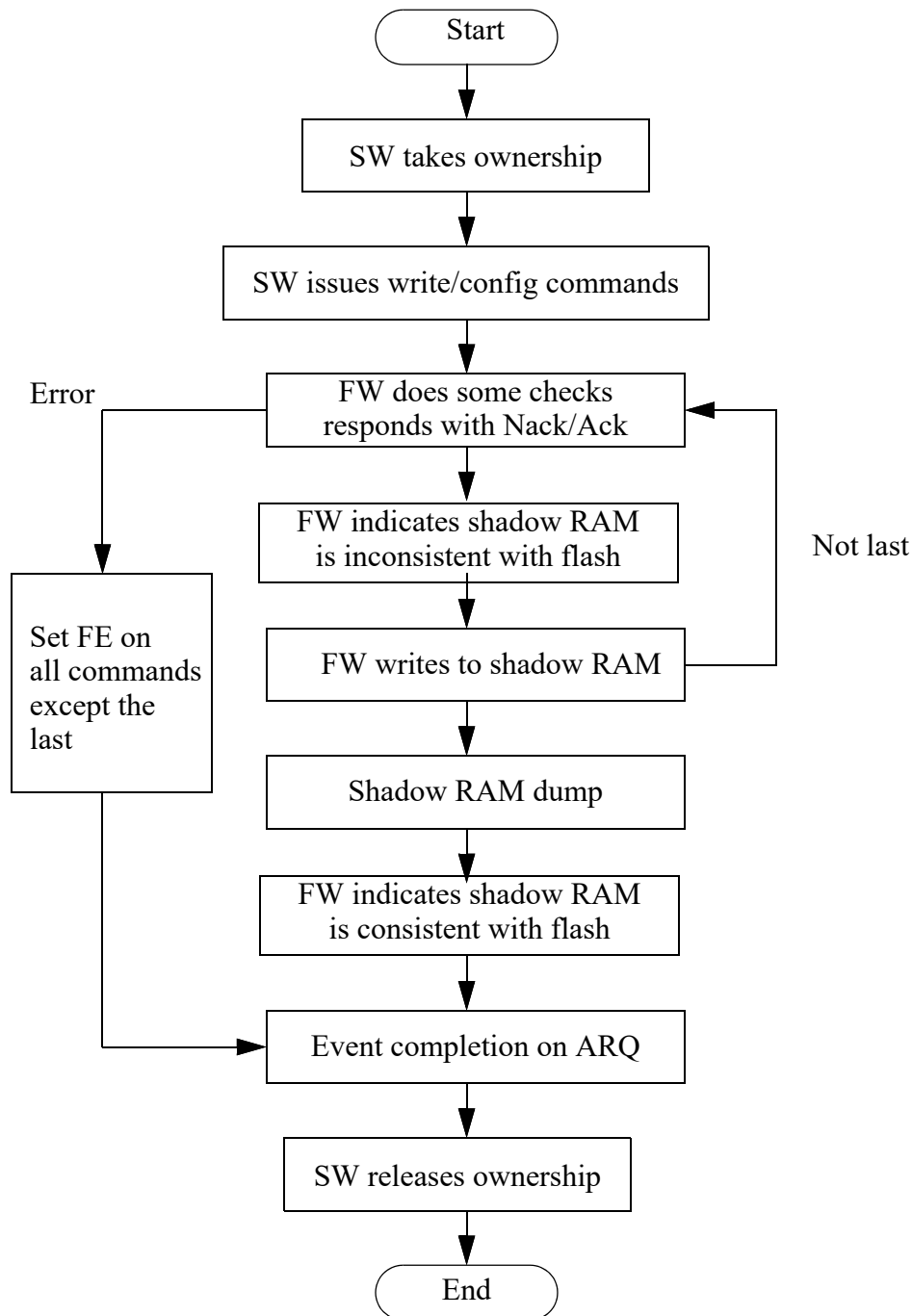


Figure 3-8. PFA Update Flow

The update flow detailed description is as follows:

1. Software takes ownership over the NVM resource for a write (see [Section 10.11.2.7.3](#)).
2. Software issues one or several NVM Write commands or NVM Config Write commands. For NVM Write admin command, see [Section 3.3.10.3](#). For NVM Config Write admin command, see [Section 3.3.10.5](#).
3. In update flows, if the *Last Command* bit is cleared in the command, it means that the command belongs to a complex NVM update operation made of several elementary NVM update commands that are posted in the admin queue. In between completions of elementary commands in a chain, other commands can be posted by a PF, besides other NVM-related commands. The entire NVM change is committed to the Flash part only once the last NVM command of the sequence is processed.

The Flush on Error (FE) bit must be set for all the commands of a sequence, with the exception of the last one.
4. The firmware checks that the command is valid and posts a response to software — It performs the following command validity checks and posts a response (ACK/NACK) to software. If one check fails then the EMP flushes the remaining NVM update commands (if any) of the sequence and exits the flow. Otherwise, if no error is encountered, the EMP runs the command (NVM update) or schedules the NVM command to run in a separate thread, resuming from the next step.
 - a. The pointer points to a section within the PFA area. The start/end offsets, once applied to the module's location in the basic bank, do not lead to addresses beyond the PFA size. In case of "dynamic NVM Update" command, the offset/length must be within the selected module.
5. The firmware sets internal bit indicating flash and shadow RAM are inconsistent.
6. The firmware writes the change into shadow RAM.
7. If the *Last Command* bit is set in the command — The firmware dumps the shadow RAM to flash (see [Section 3.3.5.2.1](#)). Otherwise, it completes the command to software and exits the flow. When the next command is posted by software the flow continues from [Step 2](#).
8. The firmware clears internal bit indicating flash and shadow RAM are inconsistent.
9. The firmware posts an event completion on ARQ to software.
 - a. If the NVM ownership timeout for write ends before reaching this step, the EMP flushes the remaining NVM update commands (if any) of the sequence, reporting a timeout error status.
10. In config write flows, the software should initiate a shadow RAM dump by sending a NVM Write Activate command with no activation bits set
11. Software releases NVM ownership (see [Section 10.11.2.7.5](#)).
12. Software initiates a reset for loading the modifications into the E610.

3.3.5.4.1 Limitations on PFA Updates/Handling

To allow seamless upgrade and downgrade updates, the following limitations and assumptions are made as how PFA can be modified.

- When new TLVs are added, they shall be added at the end of the PFA replacing part of the padding TLV.
- Firmware ignores unknown TLVs and do not treat their presence as an error.
- Reserved fields in existing TLVs shall not be used for new usages.
- The size of TLVs must not change between versions. One exception is the ANVM feature selection TLV which might expand into the next padding area.
- TLVs are never removed from PFA, even if not used anymore.

3.3.5.5 Scratch Pads Update

The following flow only supports the full module replacement without the double-bank policy. The following NVM modules are affected by this flow:

1. Firmware Scratch Pad Area (pointed by NVM word 0x50)
2. Link Topology Scratch Pad Area (pointed by NVM word 0x4B)

Figure 3-9 shows the non-authenticated module update flow.

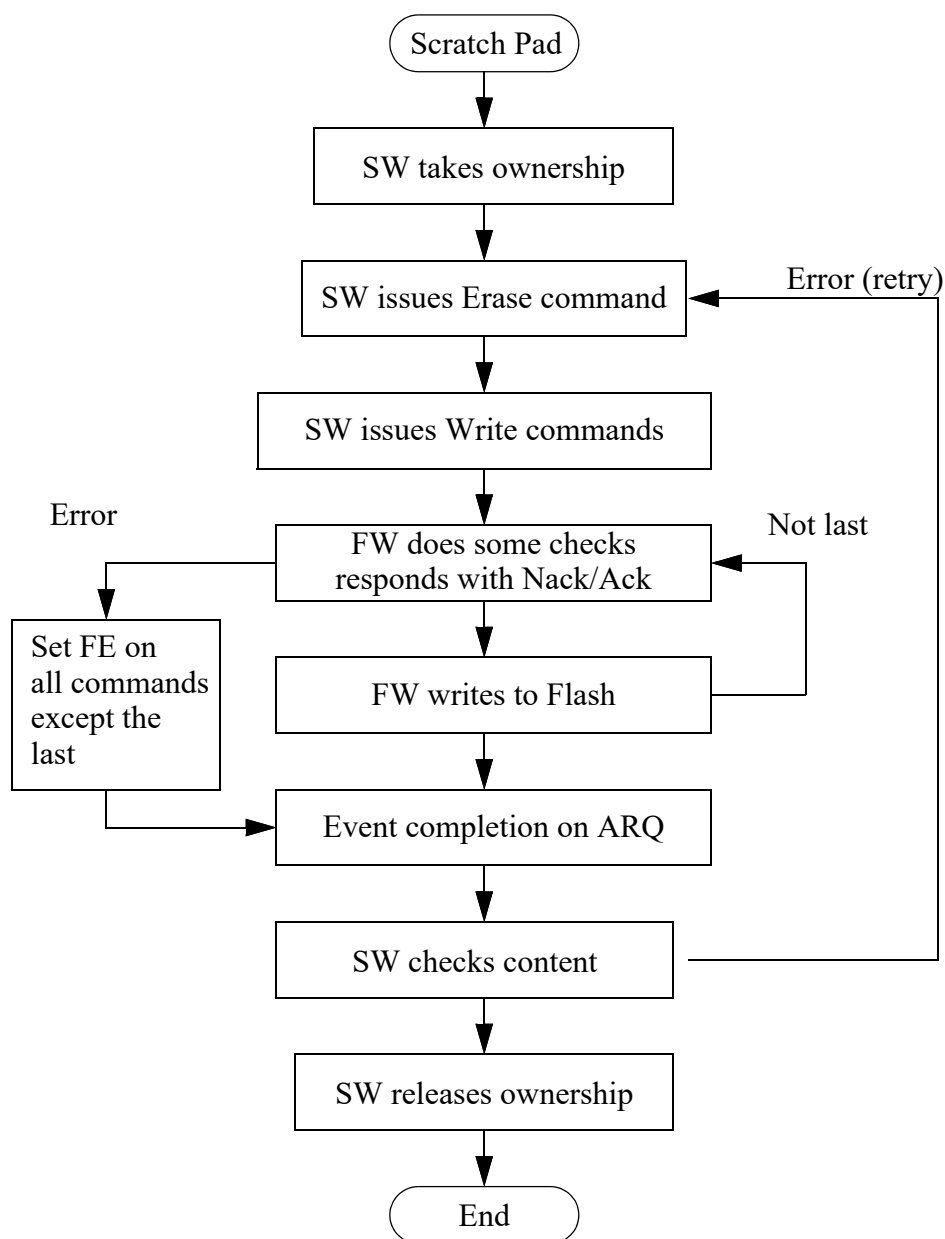


Figure 3-9. Non-Authenticated Module Update Flow

The flow details are as follows:

1. Software takes ownership over the NVM resource for a write (see [Section 10.11.2.7.3](#)).
2. Software issues a single NVM Erase command to the EMP, providing the following parameters (see [Section 3.3.10.2](#)):
 - Address in the NVM of the pointer to the relevant scratch pad area of one of the modules previously listed.
3. Software issues one or several NVM Write commands to the EMP, providing the following parameters (see [Section 3.3.10.3](#)):
 - Address in the NVM of the pointer to one of the modules previously listed.
 - Offset inside the module.
 - Buffer in host memory.
 - Buffer length (maximum supported is 4 KB).

The *Flush on Error* (FE) bit must be set for all the commands of a sequence with the exception of the last one.

4. The firmware checks that the command is valid and posts a response to software — It performs the following command validity checks and posts a response (ACK/NACK) to software. If one check fails, the firmware flushes the remaining NVM update commands (if any) of the sequence and exits the flow. Otherwise, if no error is encountered, the EMP schedules the NVM command to run in a separate thread, resuming from the next step.
 - a. The pointer location belongs to one of the modules in the list.
 - b. The offset and length, once applied to the relevant area, do not lead to addresses beyond the device's NVM area size.
 - c. The offset and length, once applied to the relevant area, do not spread over two consecutive 4 KB sectors.
5. The firmware posts an event completion on ARQ to software.
 - a. If the NVM ownership timeout for write ends before reaching this step, the EMP flushes the remaining NVM update commands (if any) of the sequence, reporting a time out error status.
6. Software releases NVM ownership (see [Section 10.11.2.7.5](#)).
7. Software takes ownership over the NVM resource for a read — refer to [Section 10.11.2.7.3](#).
8. Software reads from flash the contents of the new module to check it has been correctly recorded into the flash.
 - a. In case it was not, software redo the flow.
9. Software releases NVM ownership (see [Section 10.11.2.7.5](#)).
10. Software initiates a device reset (PCIR,) for loading the modifications into the E610.

3.3.5.6 Updating a Double-Bank Module

The following flow only supports the full bank replacement via a double-bank policy. The following banks are affected by this flow:

- **NVM** — Word 0x42 holds the 1st NVM bank pointer and word 0x43 holds the NVM bank size. The valid bank is indicated by a bit in Control Word 1. The 2nd NVM bank is right after the 1st NVM bank and has the same size.
- **OROM** — Word 0x44 holds the 1st OROM bank pointer and word 0x45 holds the OROM bank size. The valid bank is indicated by a bit in Control Word 1. The 2nd OROM bank is right after the 1st OROM bank and has the same size.
- **Netlist** — Word 0x46 holds the 1st TLV bank pointer and word 0x47 holds the netlist bank size. The valid bank is indicated by a bit in Control Word 1. The 2nd netlist bank is right after the 1st netlist bank and has the same size.

Figure 3-10 shows the double banked modules update flow:

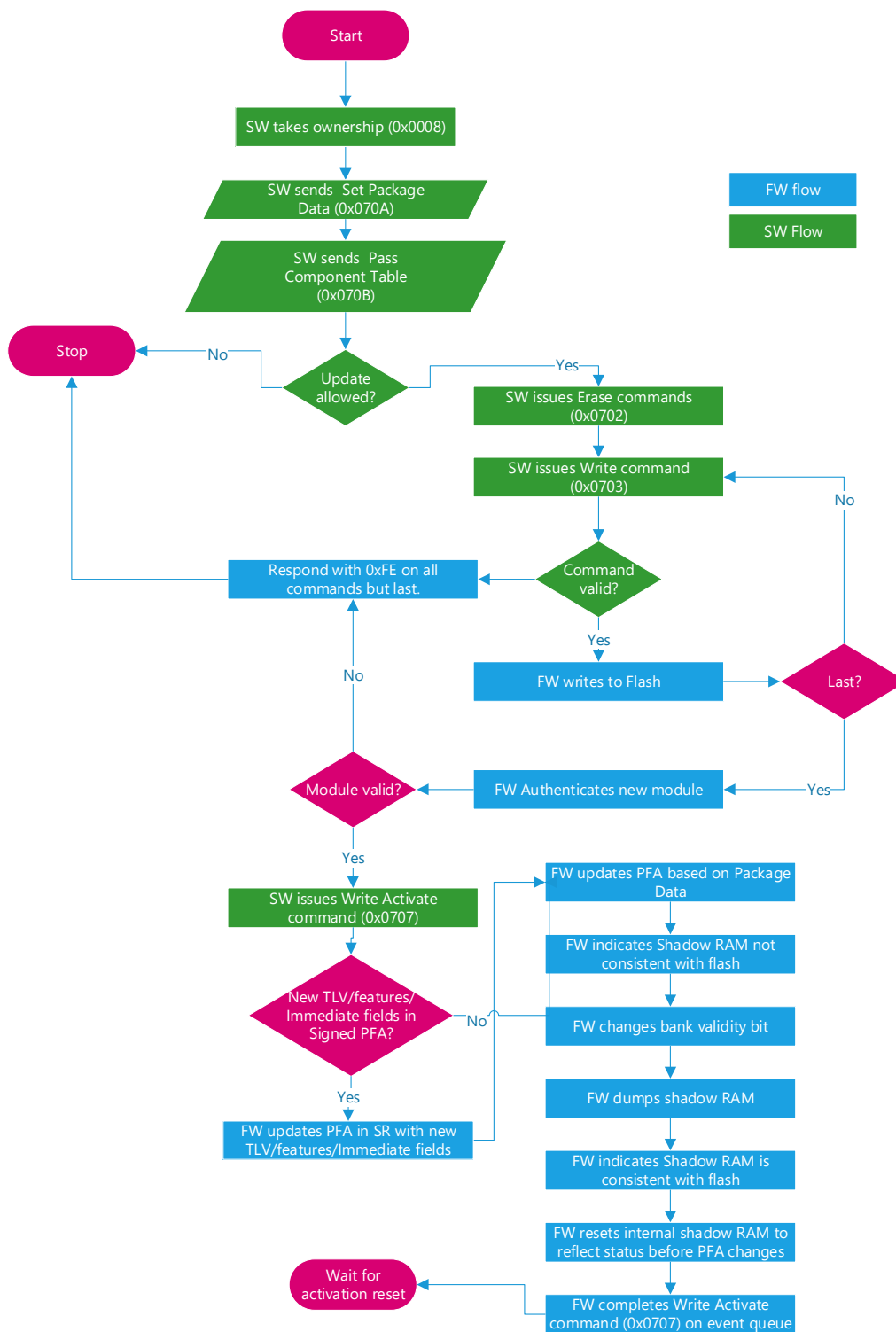


Figure 3-10. Authenticated Module Update Flow

The detailed flow is described as follows:

1. Software Tools verify the source/destination NVM images are compatible:
 - a. Takes ownership on the NVM (see [Section 10.11.2.7.3](#)) and reads valid NVM sections through admin commands (see [Section 3.3.10.1](#)).
 - b. Send a Set Package Data (0x070A) command to provide package metadata to firmware.
 - c. Send a Pass Component table (0x070B) command to check if images are compatible.
 - d. If the images are not compatible, Software Tools notifies users of the error and exits the flow.
2. Software takes ownership over the NVM resource for a write (see [Section 10.11.2.7.3](#)), if not already done so before.
3. Software issues a single NVM Erase command to the EMP, providing the following parameters (refer to [Section 3.3.10.2](#)):
 - TypeID of the module to be replaced.

Firmware knows to erase the non valid bank.

4. Software issues several NVM Write commands to the EMP, providing the following parameters (see [Section 3.3.10.3](#)):
 - TypeID of the module to be replaced.
 - Offset inside the module
 - Buffer in host memory
 - Buffer length (maximum supported is 4 KB)

If the *Last Command* bit is cleared in the command, it means that the command belongs to a complex NVM update operation made up of several elementary NVM write commands that are posted in the admin queue. In between completions of elementary commands in a chain, other commands can be posted by a PF, besides other NVM-related commands. The entire non valid bank must always be written.

The *Flush on Error* (FE) bit must be set for all commands of a sequence with the exception of the last one.

5. The firmware checks that the command is valid and posts a response to software — It performs the following command validity checks and posts a response (ACK/NACK) to software. If one check fails, the EMP flushes the remaining NVM Update commands (if any) of the sequence and then exits the flow. Otherwise, if no error is encountered, the EMP schedules the NVM command to run in a separate thread, resuming from the next step.
 - a. The module TypeID belongs to one of the banks in the list.
 - b. The offset and length, once applied to the relevant non valid bank area, do not lead to addresses beyond the non-valid bank area size.
 - c. The offset and length, once applied to the relevant non valid bank area, do not spread over two consecutive 4 KB sectors
6. The firmware writes the non valid bank area — The firmware writes the change required by the (elementary) command into it. If the last bit is cleared, the firmware posts an event completion on ARQ to software and completes the command. The flow continues from [Step 3](#) for the next (elementary) command of the sequence.

7. If Last flag is set, the firmware performs the following security checks and authenticates the next NVM bank. It reads the new bank from the flash, and authenticates its signature according to the procedure described in [Section 3.3.9](#). If one of the security check fails or if authentication fails, firmware posts an error event on ARQ to Software Tools (either the Public Key Check Fails bit or the Module Signature Check Fails), The flow goes to [Step 21](#).
 - a. The *lad_srev* in the destination is equal or greater than the one stored in the device fuses or the NVM image. The determination of whether to enforce SREV by fuses is done by checking whether the *fuse_srev_in_fuses* bit is set. The two SREV values stored in fuses relevant here are *orom_srev* (for OROM SREV value) and the *bank_srev* (for the overall bank SREV value).
 - b. The *Blank NVM Device ID* field in the destination corresponds to the E610 device ID.
 - c. The *lad_module_id* in the destination corresponds to bank currently being updated.
 - d. In case of PLDM firmware update, GFID match is checked as described in [Section 3.3.5.6.1](#).
 - e. The checks for netlist update are specified in [Section 3.3.9.3](#).

Note: Before authenticating the option ROM and netlist modules, the module must be appended with the last 842 words of the module's area, as the CSS header has been moved to the module's trailer. Refer to [Section 6.1.5.2](#) for more details.
8. Firmware posts an event completion on ARQ to software.
 - a. Where applicable, software driver shall save *RESET_FLAG* as returned by this command.
9. Software Driver may release NVM ownership, or keep it until NVM Write Activate is completed.

Note: An attempt to write to a ready double bank at this stage will result in rollback of the previous write so that authentication will need to be done again on new write completion.
10. Write command which modify PFA TLV configuration (including VPD) may be called with Last bit cleared - to activate (dump to flash) in the same time of the Bank Update Activation command. However, it is recommended to use the Package Data provided in Get Package Data command to request PFA modifications.
11. If not already taken, software driver takes ownership over the NVM resource for a write.
12. Software driver sends an NVM Write Activate Command and states which modules need activation by setting a flag per module (NVM Bank, OROM, Topology Netlist). Shadow RAM Bank will be written by default and does not require setting any flag. Only banks that were written completely (i.e. Last bit was set) can be activated. Firmware will post a fail response to the command if one of the following checks fails:
 - a. All specified banks were written.
13. If a NVM bank was written, it is also activated. Firmware sets internal bit indicating flash and shadow RAM are inconsistent.
14. Firmware parses the content of the package data provided by the Set Package Data command and modifies the PFA accordingly.
15. Depending on preservation mode, firmware overwrites current PFA with the default PFA from the image (no preservation), replaces selected PFA fields in current PFA with the default ones from the image (selective preservation), copies the factory setting region to the PFA (restore to factory settings) or makes no changes to current PFA (full preservation).

16. In case of NVM module update, firmware compares the TLVs present in signed version of the PFA with the ones in the actual PFA. If a TLV is present in the signed PLA and not in the actual PFA, it is added to the actual PFA replacing part of the padding TLV. It also checks if new features or new immediate fields are added to the PFA version of the signed NVM. If there are new features or immediate fields they are copied with their default values to the PFA in the shadow RAM.

- a. If features changed, the firmware will run the ANVM flow to make sure any affected PFA TLVs are updated.

Note: From this point until initiating the reset that will switch to the new firmware, NVM Write (0x703) to PFA or to any of the double Banks will be rejected.

17. The firmware changes the valid bit indication in Control Word 1.

18. The firmware dumps the shadow RAM to flash (see [Section 3.3.5.2.1](#)).

19. Firmware clears internal bit indicating flash and shadow RAM are inconsistent.

20. The firmware posts an event completion on ARQ to software — It posts a completion/response to the last admin command of the sequence.

21. Software releases NVM ownership. Refer to [Section 10.11.2.7.5](#).

22. At this stage, if a new update is needed, the software may revert to previous state by using the NVM Write Active command with revert flag set. After that it can restart the flow for a new image.

23. Software checks the *RESET_FLAG*.

24. **Software Tools** checks the value of the *RESET_FLAG* field in the event posted by firmware:

- a. If *RESET_FLAG* == PERST: Software Tools notifies the user to reboot the system since Software Tools have no ability to issue a PERST cycle (it can only ask the OS to do so).
- b. If *RESET_FLAG* == POR: Software Tools notifies the users to reboot the system (to initiate a PERST cycle) as before.

Note: In PLDM firmware update case, firmware sends a feedback to BMC with the requested reset to activate the update.

25. Software may choose to activate the new firmware in two ways:

- a. **Wait for the next PERST event:** On the next PERST event, firmware also resets itself by issuing an EMPR cycle. In this case, the PERST may take longer time to complete and may violate PCIe rules.
- b. Send a NVM Update EMPR command, and then request for an immediate PERST event. In this case, the NVM Update EMPR command will create an EMPR, and the PERST will be faster and compliant with the PCIe rules. **The software should not initiate any NVM change between the two resets.**

26. **Autoload (driven by new firmware)** occurs from the new SR contents.

27. **Software Tools** checks that the destination map *EETrackID* was loaded in Internal SR to verify we did not have spurious reset during update, causing ROM to load wrong shadow RAM bank.

- a. In case the original map '*EETrackID*' is read from Internal SR, Software Tools restarts the flow from the beginning.

28. **(New) Firmware** loads the new firmware code and other NVM settings into its Internal RAM.

Note: Contents of the old module cannot be erased by a software command prior to completing this flow.

3.3.5.6.1 GFID Handling

GFID is used to identify if two images are compatible. If the GFID of the current image is the same as the GFID of the new image, then the new image can be used to upgrade the current one. In some cases, it is needed to allow a split of a single family of images to multiple families. As GFIDs must be defined as part of the first image released, a mechanism to move from one GFID to another is needed. Hence each image contains 2 GFIDs. An original GFID and a current one.

Figure 3-11 shows the algorithm used to define if an image can be updated based on the two GFIDs in the current (Adapter) and next (Update) images.

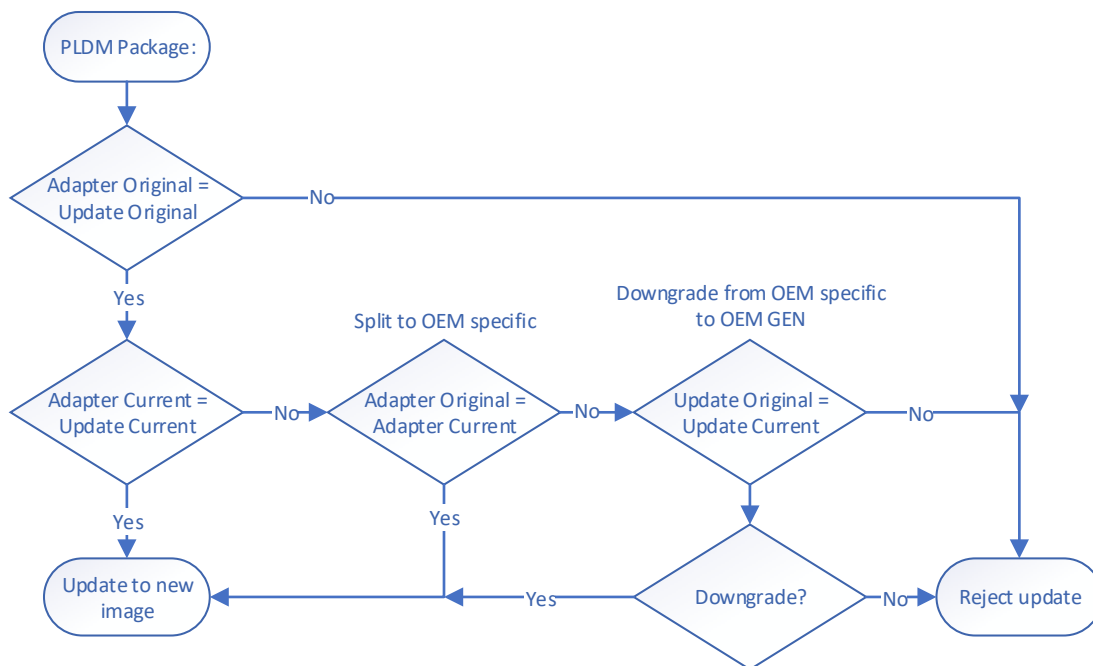


Figure 3-11. GFID Handling Flow

The GFIDs are stored in the non PFA part of the shadow RAM for the current image and in the PackageData of the PLDM header for the new image.

3.3.5.6.2 Recovery from Reset During Flash Update

In case a spurious reset occurs (other than POR) before the update sequence is fully completed, the following flow is used to recover:

1. Firmware issues a graceful EMPR cycle, notifying Software Tools about the event.
2. Software Tools report to user that the update has failed. Software is notified by interrupt on the fact the firmware was reset.
3. User restarts the flow from the beginning.

3.3.5.7 Flash Wear-Out Protection

The mechanism prevents flash part wearing out by malicious software. This mechanism blocks excessive flash erase operations.

Assuming a typical flash device endurance is ~100,000 erase/program cycles and the required system lifespan is 10 years, a flash erase can happen every $10 \text{ (years)} * 365 \text{ (days/year)} * 24 \text{ (hours)} * 60 \text{ (minutes)} / 100,000 = \sim 53 \text{ minutes}$.

The E610 does not have access to the real time clock and its clocks do not survive a POR reset. As a result the algorithm have some limitations related to POR resets. Assuming a POR is not a frequent event which can not be initiated by malicious software it does not pose a problem.

Note: An entry to the Sx power state is equivalent to power on reset for the systems with no AUX power.

The simple algorithm described in the figure below is credit based. Separate credits are counted for every erase unit. To avoid credit counter proliferation the number of erase units is limited to the following blocks, while more blocks can be added in future if there is such a need:

- Two NVM Banks
- Two OROM Banks
- Two persistent SR Banks
- Two TLV Extension Banks
- Link topology Scratch Pad Area
- Firmware Scratch Pad Area

It is noted that limiting the number of units requires modification of the NVM update flows to allow only the whole unit erase at a time blocking the per sector erase operations. To simplify system validation and flash content update after the first power on of the system there are initial credits assigned to each unit.

The recommended values for the initial and maximal credits are 2000 (2%) and 5000 (5%) correspondingly. These values are stored in the signed part of the NVM image. The algorithm is presented in [Figure 3-12](#).

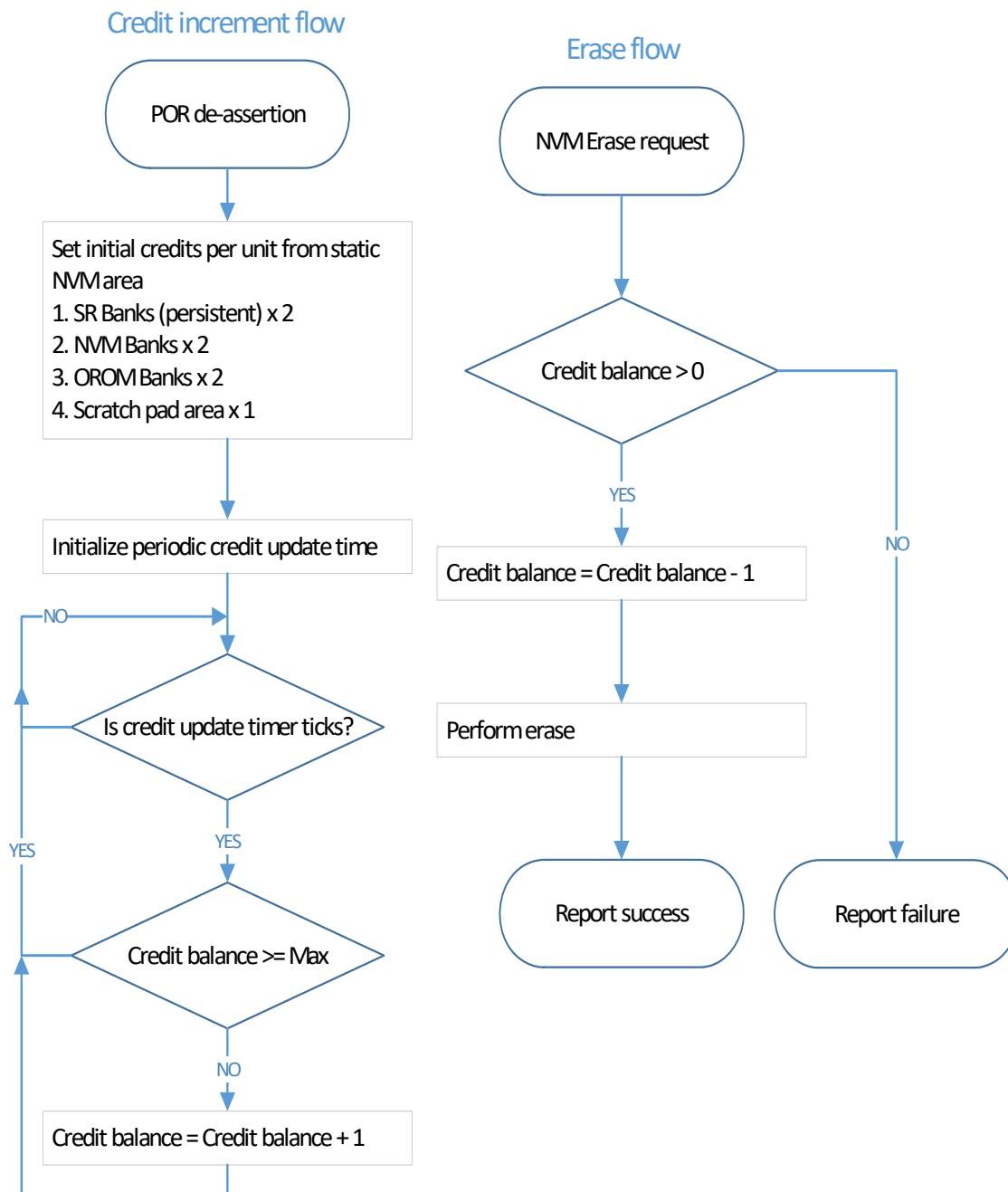


Figure 3-12. Flash Wear-Out Protection Algorithm¹

1. Credit update time is once per hour.

3.3.5.8 Save Factory Settings AQC

This section describes the flow executed by the “Save Factory Settings (0x0708)” on page 131.

3.3.5.8.1 Save Factory Settings Flow

The following flow is run by the “Save Factory Settings” admin command (opcode 0x708) and can be used by manufacturing tools to modify the image before programming the device.

- Word pointer to PFA in internal Shadow RM (iSR) is located at iSR word offset 0x40.
- PFA word size is at first PFA word.
- Word pointer to Netlist is located at iSR word offset 0x46.
- Netlist word size is at 2nd netlist word.
- Word pointer to Factory Settings is located at iSR word offset 0x61.

Flow:

1. Verify PFA checksum.
2. Check that Factory Settings (FS) password is “0xFFFF”.
3. Setup a variable to keep FS size within limits in init module word offset 0x62.
4. Copy PFA content from iSR to Factory settings PFA area (offset 32B from address in word 0x61).
5. Copy Netlist content from valid Netlist bank to Factory settings Netlist area (offset 32B + PFA size from address in word 0x61).
6. Set FS header fields located at the first 32 byte of Factory settings:
 - password = 0x2811
 - Size = 32B + PFA size + netlist size
 - TLV_Extension_Offset = 32B + PFA size
 - TLV_Extension_Size = netlist size
 - PCIR_AL_Offset = PCIR AL section byte offset relative to PFA start. TLV 0x133.
 - POR_AL_Offset = POR AL section byte offset relative to PFA start. TLV 0x118.
 - PCI_Serial_ID_MAC_Offset = PCI Serial ID MAC section byte offset relative to PFA start. TLV 0x133.
 - Set all reserved bytes to '0'.

3.3.6 NVM Clients and Low-Level Interfaces

There are several clients that can access the NVM to different address ranges via different access modes, methods, and low-level interfaces. The various clients to the NVM are hardware, software tools (BIOS, etc.), drivers, EMP, PE, BMC (via EMP), and VPD software.

Table 3-21 lists the different accesses to the NVM.

Table 3-21. Clients and Access Types to the NVM

Client	NVM Access Method	Access Performed Against	Logical Byte Address Range	NVM Access Interface (CSRs or Other)
VPD Software	Parallel (32-bits)	Shadow Ram	0x000000 - 0x0003FF from VPD module beginning	VPD Address and Data registers. Any write access is immediately pushed by the E610 into the Flash.
PF Software	Parallel via Memory (CSR)BAR (32-bits read)	Flash Part	0x000000 - 0x00FFFF	The address is relative to the beginning of the flash. See Section 9.2.2.16 for details of the flash offset within the CSR BAR.
	Parallel via option ROM BAR (32 bits read only).	Flash Part	0x020000 - 0xFFFF	This logical address range is relative to the beginning of the option ROM module.
	Via AQC.	Flash Part / Shadow RAM	0x000000 - 0x00FFFF	NVM Read, NVM Erase and NVM Write admin commands as described in Section 3.3.10 .

3.3.6.1 Memory-Mapped Host Interface

The Flash is read by the E610 each time the host CPU performs a read operation to a memory location that is within the Flash address mapping or upon boot via accesses in the space indicated by the option ROM Base Address register. Accesses to the Flash are based on a direct decode of CPU accesses to a memory window defined in either:

- Memory CSR + Flash Base Address register (PCIe Control register at offset 0x10). The Flash address space is exposed to the host memory BAR when the Flash Expose bit is set in the NVM (or the GLPCI_LBARCTRL.FLASH_EXPOSE CSR bit is set) or when the E610 is in blank Flash programming mode. The Flash size exposed is configurable. Refer to [Section 9-10](#) for more details.
- The option ROM Base Address register (PCIe Control register at offset 0x30). The OROM module address space is exposed to the option ROM BAR if the Flash is not blank and the expansion ROM base address register contains a valid (non-zero) base memory address, otherwise the BAR is not exposed. The Flash size exposed is retrieved from the GLPCI_LBARCTRL.EXROM_SIZE CSR field, and is 1MB by default. The E610 is responsible to map read accesses via the option ROM BAR to the physical NVM. Write attempts to the Flash through this BAR are not performed, they are silently dropped. The offset in the NVM of the option ROM module is defined by the PCIe expansion/option ROM pointer (Flash word address 0x05). This pointer is loaded by the E610 from the Flash before enabling any access to the option ROM memory space.
 - When modifying the PXE driver section pointer in the NVM, it is required to issue a PCIe reset on which the updated offset is sampled by hardware.
 - If there is no valid NVM validity field in the two basic banks, then the option ROM BAR is disabled.

The E610 controls accesses to the Flash when it decodes a valid access. Out of range memory-mapped read access returns arbitrary data.

Notes:

- Refer to [Section 3.1.2.2.1](#) for details on memory/option ROM BAR access rules.
- Flash read accesses are assembled by the E610 each time the access is greater than a byte-wide access.
- E610 byte reads to the Flash take about 2 to 30 μ s. The E610 continues to issue retry accesses during this time.
- During normal operation, the host avoids memory-mapped accesses to the first two basic banks of the Flash because it might be non-coherent with the shadow RAM contents.

Prior to initiating an NVM read cycle via memory mapped access, PF software is required to take ownership over the NVM resources. Refer to [Section 10.11.2.7.3](#).

3.3.7 Flash Access Contention

Flash read accesses initiated through PFs might occur concurrently to the firmware modifying the NVM contents. The E610 does not synchronize between the different entities accessing the Flash so contentions caused from one entity reading and the other modifying the same locations is possible.

To avoid such a contention between software and firmware accesses, these entities are required to make use of the NVM ownership taking/release flows for any read or write access to NVM. Refer to [Section 10.11.2.7.3](#) and to [Section 10.11.2.7.5](#) for more details. This is also useful to avoid the timeout of the PCIe transaction made to a memory mapped Flash address while the Flash is currently busy with a long sector erase operation.

However, two software entities cannot use the NVM ownership acquiring/release mechanisms: BIOS reading through Expansion ROM BAR and VPD software. A BMC requesting firmware update via PLDM firmware update also does not require a semaphore.

- Since VPD software accesses only the VPD module, which is located in the first valid bank of the NVM, VPD accesses are always performed against the shadow RAM first. In this case, the firmware must take/release ownership over the NVM as if it was the originator of the Flash access. It is then hardware/firmware's responsibility to update the NVM according to the Flash update sequence described in [Section 3.3.5.3](#).
- No contention can occur between BIOS and any other software entity (VPD included) as it accesses the NVM while the operating system is down.
- The firmware will take care of the semaphore ownership for PLDM firmware updates.

However, since BIOS cannot take ownership over the NVM resource, it might be that the Flash part is not accessible when BIOS attempts reading it. This might occur if a Flash erase operation was performed just before PCIe reset. In such a case, read accesses via the option ROM BAR returns 0xDEADBEEF.

- It is assumed that the option ROM signature check performed by BIOS fails in this case.
- The firmware must avoid initiating sector erase operations at boot time.
- It is assumed and recommended that users do not attempt to update the NVM contents via the BMC while the system is re-booting.
- The BMC delays PERST# de-assertion or boot running until after the BMC completed any OOB accesses to Flash memory. It is required to route the wake-up signal from the standby button to the BMC and not to the chipset. The BMC issues a system reboot signal to the chipset only after any NVM write access completes.

- If a system reboot is issued by a local user running on the host, there is no technical way to avoid contention in this case.

Note: It is the user's responsibility when accessing the NVM remotely via the BMC to make sure another user is not currently initiating a local host reboot there.

- The firmware is responsible to take NVM ownership on the BMC account prior to performing any NVM read or write access, which is needed for handling an NC-SI command. The NVM ownership is released by the firmware together with completing the NC-SI command. If NVM ownership is not free when processing the NC-SI command, the command completes with a package not ready status.

3.3.8 NVM Access Procedures

Any software read/write or firmware write flow described in this section (except flows executed by VPD software or by BIOS or to flows executed when in blank Flash programming mode) must be preceded by taking NVM ownership. Anytime software is taking NVM ownership, it must re-read the pointers to the module it plans to access because they might have been modified by the firmware in between two ownership takings.

Refer to [Section 10.11.2.7.3](#) and [Section 10.11.2.7.5](#) for the NVM ownership taking/releasing procedures as well for the associated timeouts.

3.3.8.1 Autoload

This section describes only the autoload/init tasks relative to NVM. Firmware is solely responsible for all autoloads and initialization flows described in this section.

Firmware initialization performed at power up and EMPR events consists for three steps, while DIGEST updating documented in 3.6.8.1.4 is done at each step:

1. ROM flow:
 - a. Find a valid NVM content.
 - b. Load the Extended mini-loader to the local RAM and authenticate it.
 - c. Jump to the mini-loader code entry point.
2. Mini-loader flow:
 - a. Autoload time critical NVM section, which a part of the Extended mini-loader and the persistent Shadow RAM bank. Starting from this point mini-loader is ready to handle time critical PCIe auto-loads.
 - b. Read the shadow RAM content to the local RAM from the NVM.
 - c. Read the whole NVM bank to the local RAM and authenticate it. The EMP firmware is placed in the local RAM at this moment, while most of the other content is discarded after the authentication.
 - d. Jump to the EMP firmware entry point.
3. EMP firmware flow:
 - a. Perform autoload on every PCIe, resets.
 - b. Initialize all firmware code being ready to handling all run-time events.
 - c. Report initialization completion to the host software.

For better understanding of the initialization flows, simplified NVM Bank and Extended mini-Loader maps are shown in [Figure 3-13](#).

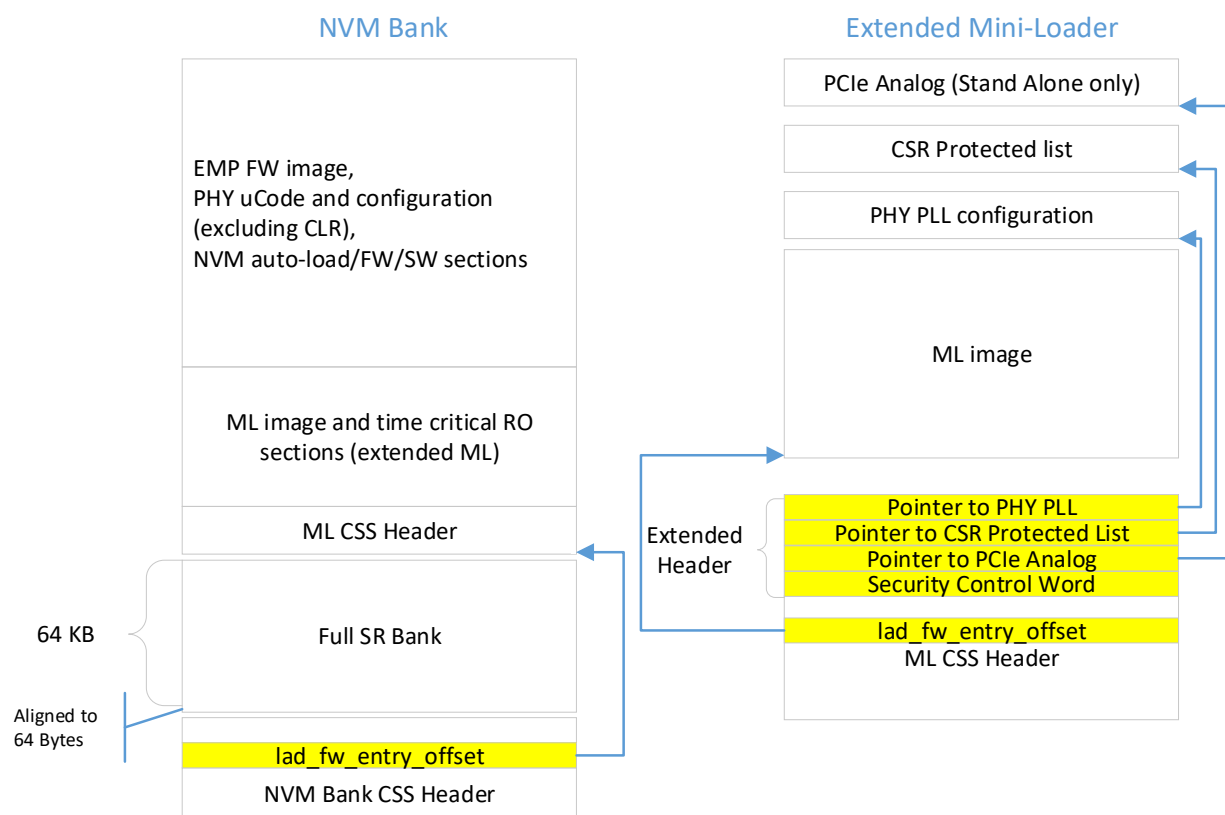


Figure 3-13. Simplified NVM Bank and Extended Mini-Loader Maps

3.3.8.1.1 EMP Firmware Init Flow

The firmware is ready to PERST de-assertion handling during the flow. Such an event must be handled within 20 ms.

1. Run the adaptive NVM initialization flow using feature selections in PFA on the shadow RAM copy in RAM. No shadow RAM dump is required since the same initialization occurs after every POR reset.
2. Proceed initialization firmware initialization.

3.3.8.1.2 Firmware Measurement for Remote Attestation

The EMP firmware is required to expose firmware measurement to the host software. This measurement is represented by a cryptographic identity of all firmware components. Two mechanisms are supported in the E610:

- Three sets of CSRs expose measurements to the host software via the PCIe configuration space.
- Device attestation via the DMTF SPDM protocol is also supported, for details.

Three-byte DIGEST registers (GL_MNG_SHA_EXTEND – [Section 8.2.2.23.6](#), GL_MNG_SHA_EXTEND_ROM – [Section 8.2.2.23.11](#)) and GL_MNG_SHA_EXTEND_COMBINED – [Section](#)

TBD are implemented in the E610 for this purpose. Another GL_MNG_SHA_EXTEND_STATUS register (Section 8.2.2.23.4) exposes the measurement status.

Note: The GL_MNG_SHA_EXTEND_ROM, GL_MNG_SHA_EXTEND, and GL_MNG_SHA_EXTENR_COMBINED are writable only once after each EMP reset.

CSR firmware attestation flow:

1. The E610 ROM locates the E610 mini-loader.
 - a. If the mini-loader is not found¹, the E610 ROM sets GL_MNG_SHA_EXTEND_ROM to the ROM version, sets GL_MNG_SHA_EXTEND_STATUS.STAGE to 001, sets GL_MNG_SHA_EXTEND_STATUS.FW_HALTED, sets GL_MNG_SHA_EXTEND_STATUS.DONE, then moves to blank flash programming.
2. If mini-loader is found, the mini-loader header contains a SHA hash of the mini-loader and signature over the hash. The E610 ROM authenticates mini-loader.
 - a. If the authentication is successful, the E610 ROM writes the mini-loader hash value to GL_MNG_SHA_EXTEND_ROM register and sets GL_MNG_SHA_EXTEND_STATUS.STAGE to 010. In this case, the computed mini-load hash matches what is in the header, the ROM loads the mini-loader and passes control to it.
 - b. If the authentication fails, the E610 ROM sets GL_MNG_SHA_EXTEND_STATUS.STAGE to 001, sets GL_MNG_SHA_EXTEND_STATUS.FW_HALTED, sets GL_MNG_SHA_EXTEND_STATUS.DONE, then halts. In such cases, the computed hash may not be a reliable measurement and therefore the value in the GL_MNG_SHA_EXTEND_ROM register will be the ROM version.
3. The E610 mini-loader locates the E610 firmware header. If the firmware is not found, the E610 mini-loader sets GL_MNG_SHA_EXTEND_STATUS.STAGE to 011, sets GL_MNG_SHA_EXTEND_STATUS.FW_HALTED, sets GL_MNG_SHA_EXTEND_STATUS.DONE, then moves to blank flash programming.
4. NVM Bank header contains a SHA hash of the NVMBank content. Note that the E610 mini-loader is itself contained within the NVM Bank. The E610 mini-loader authenticates the complete NVMBank content.
 - a. If the authentication is successful, the E610 mini-loader writes the NVMBank hash value to GL_MNG_SHA_EXTEND register. In addition, the mini-loader programs the combined hash value into GL_MNG_SHA_EXTEND_COMBINED register. The combined measurement is the hash of hashes: COMBINED = SHA384(CONCAT(SHA384(ML), SHA384(NVM Bank)), where CONCAT is the concatenation function. In this case, the computed NVMBank hash matches what is in the header, the E610 mini-loader sets:
 - GL_MNG_SHA_EXTEND_STATUS.STAGE to 100
 - GL_MNG_SHA_EXTEND_STATUS.DONE
 - b. If the authentication fails, the E610 hash of the image may not be a reliable measurement and the value in the GL_MNG_SHA_EXTEND register may be zero and should not be used. In such cases, the E610 mini-loader sets the following, then halts:
 - GL_MNG_SHA_EXTEND_STATUS.STAGE to 011
 - GL_MNG_SHA_EXTEND_STATUS.FW_HALTED
 - GL_MNG_SHA_EXTEND_STATUS.DONE

1. If the RSA key verification fails this is considered as if mini-loader is not found.

Software flow:

1. Poll the GL_MNG_SHA_EXTEND_STATUS.DONE until set.
2. Keep value of GL_MNG_SHA_EXTEND_STATUS
3. Read the GL_MNG_SHA_EXTEND and GL_MNG_SHA_EXTEND_STATUS registers.
4. Reread the GL_MNG_SHA_EXTEND_STATUS register and check the value is the same as saved. If not, the firmware may have been reset during the read process.
Note: The complete read flow should complete within 100 ms to guarantee that a full reset cycle did not occur in between.
5. If GL_MNG_SHA_EXTEND_STATUS.FW_HALTED, the firmware load did not succeed.
6. The GL_MNG_SHA_EXTEND_STATUS.STAGE indicate the latest stage of the firmware authenticated.

3.3.8.2 VPD Access

The VPD module (VPD area) is mapped into the valid basic bank and it is thus mirrored in shadow RAM. It is accessed by VPD software via the PCIe VPD capability structure. Refer to [Section 3.3.5.3](#) for more details.

3.3.9 NVM Authentication Procedure

The NVM update integrity feature ensures that only Intel-approved firmware code (or another protected NVM module) is able to run on E610 devices after manufacturing. This procedure is performed by mini-loader and ROM-code on every initial program load, and by a RAM-based firmware each time an attempt is made to update one of the protected modules. Refer to NVM update flows in [Section 3.3.5](#) for more details. The OROM image is not authenticated by EMP on device boot. It is authenticated only on OROM updates under the assumption that the system BIOS will authenticate the OROM prior to executing it during platform boot.

Integrity validation of NVM updates is provided by means of a digital signature. The digital signature is a SHA2-384 hash computed over the protected content that is then encrypted by a 3072-bit RSA encryption using an Intel private key. This digital signature is stored in what is called the manifest in the NVM module image. Also stored in the manifest is the corresponding RSA modulus (public key) and RSA exponent to be used to verify the digital signature. Refer to [Section 6.1.5](#) for more details.

To verify the authenticity of the digital signature of any protected content, the EMP ROM must first verify that the SHA2-512 hash of the RSA Modulus, RSA Key Prime, RSA Key Second and RSA Exponent fields in the new module loaded are identical to those retrieved from a trusted source. If the hash of the RSA Modulus, RSA Key Prime, RSA Key Second and Exponent fields match the value in the trusted source, EMP authenticates the digital signature using the 3072-bit RSA Modulus, Key Prime, Key Second and Exponent fields stored in the manifest. Once verified it extracts the expected SHA2-384 hash of content (stored hash). EMP then performs an independent SHA2-384 hash over the protected content (computed hash). If the stored hash matches the computed hash, the digital signature is accepted and the protected content is verified as legitimate.

This process is illustrated in [Figure 3-14](#), which indicates the areas within the manifest which are hashed and signed. Note that the hash algorithm for generating the hash of the protected source and the decryption algorithm applied to the digital signature are both defined in the CSS header. The illustration shows the typical case where it is configured for SHA2-384 and RSA 3k. The hash of the public key elements always uses SHA2-512 and is not configurable.

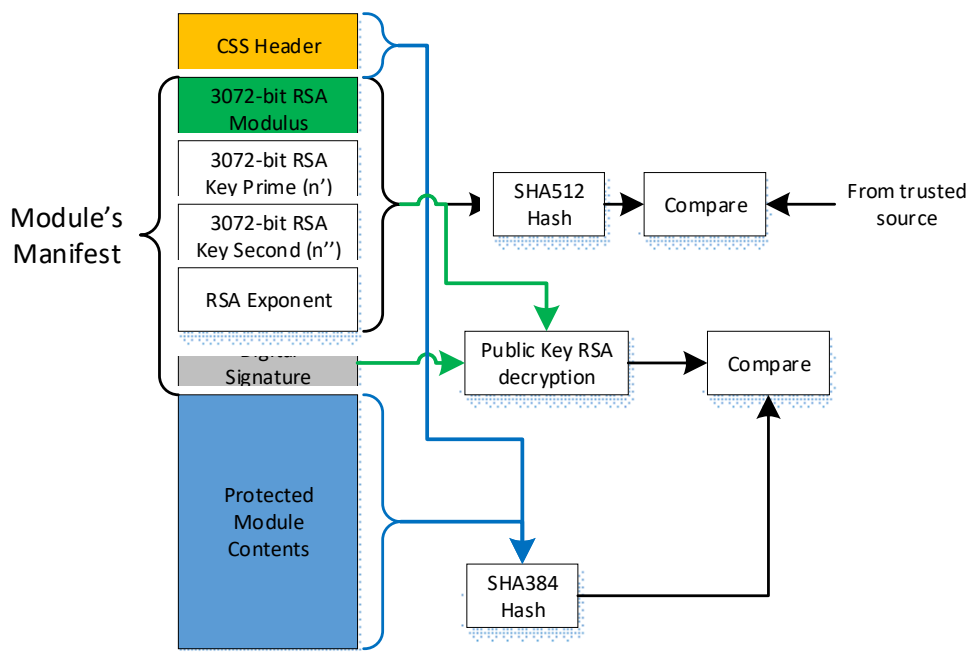


Figure 3-14. Verify Procedure for Authenticated Modules

The trusted sources vary by the type of protected content verified:

1. The Key Authentication Key hashes are stored in the device fuses. The KAKs are used to verify the integrity of the Image Signing Keys. The ISKs, in turn, are then used to verify additional NVM content.
2. The Image Signing Key hashes are stored in the NVM, and are signed by the KAKs. The ISKs are used to sign the other three types of protected content. A different ISK is used for each type of E610 Ethernet Controller — Interconnects content, and each ISK Object also includes an ID field (also signed by the KAK) which defines the revocation fuse index associated with the key to allow for fuse-based key revocation. The following protected content is signed by ISKs stored in the NVM:
 - a. NVM Bank (including mini-loader, Recovery FW, Signature Key Revocation Module) — The NVM Bank contains most of the configuration, including the EMP firmware itself, required for the device to operate. The NVM Bank is authenticated on every boot and during NVM update.
 - b. Option ROM — The OROM is a pre-boot executable loaded by the system BIOS from the device NVM and executed on the host platform. The OROM is authenticated by the device during NVM update.
 - c. Link Topology Netlist — The Link Topology Netlist describes how internal or external components are connected to the Ethernet Link and how to access them in order to configure them.
 - d. OEM RoT Key Revocation Module - The Key Revocation Module identifies which of the KAKs or ISKs are allowed to be revoked.

NVM updates are validated prior to invalidating the old NVM configuration, such that the old NVM configuration is still usable if the update fails to validate. After the new NVM is successfully verified, the updated image is committed to the E610's Flash by the EMP.

3.3.9.1 Digital Signature Algorithm Details

As previously mentioned, the digital signature generation is a hash computation followed by an RSA signature. This is performed within Intel as part of the NVM update image generation process and not performed by Intel software in the field, nor by the E610.

The algorithms used are described in the following locations:

- PKCS #1 v2.2: encoding method - EMSA-PKCS1-v1_5: <https://tools.ietf.org/html/rfc8017>
- SHA family definition - <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>
- SHA usage with digital signatures - <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-107r1.pdf>
- SHA validation vectors — <http://csrc.nist.gov/groups/STM/cavp/documents/shs/SHAVS.pdf>

Note: The protected module contents shown in Figure 3-14 starts with the CSS header, skips the RSA Public key, RSA exponent and Encrypted SHA hash, continues from the E610's Blank NVM Device ID word of the NVM header described in Section 6.1.5.2 and ends at the offset defined by the size field in the CSS header.

3.3.9.2 Intel Key Generation and Intel Code Signing System

The integrity of NVM digital signatures requires not only robust private RSA key generation but also continued protection of these private keys into the indefinite future as well as a method to generate new signed images using the existing private keys.

Intel's CSS follows the PKCSv2.2 format with 3072-bit RSA keys and SHA2-384 hash - RSA-PSS. The CSS algorithm requires a standard manifest header to appear at the beginning of all signed modules.

The manifest header (represented in C syntax) is as follows:

```
typedef struct _CssHeader {
    uint32 moduleType;           // Reserved CSS field
    uint32 headerLen;           // Reserved CSS field
    uint32 headerVersion;       // Reserved CSS field
    uint32 moduleID;            // Reserved CSS field
    uint32 moduleVendor;        // Reserved CSS field
    uint32 date;                // Reserved CSS field
    uint32 size;                // In DWords - Reserved CSS field
    uint32 keySize;             // Reserved CSS field
    uint32 modulusSize;         // Reserved CSS field
    uint32 exponentSize;        // Reserved CSS field
    uint32 lad_srev;            // Internal security revision field
    uint32 crypto_info;         // Concatanation of {uint4 reserved, uint4 srev_flags,
                                //uint8 encryption_algorithm, uint8 hash_algorithm, uint8
                                // signing_algorithm}
    uint32 lad_fw_entry_offset;  // ND-specific offset from start of CSS header of the
                                // firmware code in WORDS.
    int32 reserved;             // Mandatory field, in use by project
    uint32 lad_image_unique_id; // Internal unique identifier for the
                                // specific NVM image version. Taken from NVM Software
                                // Reserved Words 22, 23 (a.k.a. EETRACK ID). N/A for
                                // mini-loader CSS header.
    uint32 lad_module_id;       // Internal and per device unique
                                // identifier for NVM module.
}
```

```

// Is one of the following values:
//   * 0x5 for Option ROM
//   * 0x6 NVM Bank
uint32 reserved[16]; // Mandatory field, but free for use
} CssHeader;

```

The CSS header must be placed at the beginning of the integrity-checked data. Software tools are also required to pre-process raw NVM images to prepend the manifest prior to CSS tool chain submission. All fields marked *Reserved CSS Field* must be zero when submitting to the CSS tool chain.

The E610 CSS header includes a Security Revision (*lad_srev*) field. This field is monotonically updated for each and every security-related update to the NVM and ranges from the special value of zero (detailed below) to 0xFFFFFFFF (that is, 32 bits are allocated for each srev). If a security exploit is detected and an updated NVM image is released with an incremental security revision, the E610 does not allow an NVM image roll back to an earlier version because this might expose previously known vulnerabilities.

The security version checking always takes place during NVM Update. If the *srev_in_fuses* fuse is blown, then srev checking will also take place as part of the ROM and ML flows, and the srev value will be stored in fuses. There is no ability to roll back to an image with an older security revision if the srev value is stored in fuses, and even physical programming of the device's flash will not allow rollback. SREV values in fuses use a unary encoding. Thus, if the least significant bit of the fuse field is set, this represents a value of 1. If the two least significant bits are set it represents 2, etc. The device converts between formats in hardware when performing the comparison.

If the fuse is not blown, then the ROM and ML will not enforce srev checking, allowing the part to be reprogrammed to an older NVM image via physical access.

Security version checking can only be temporarily disabled by a specific manageability command. (See TBD) under the following conditions:

- The *srev_in_fuses* fuse is not blown.
- The NVM configuration allows for this command to be received by the device.

If the srev of the signed image is zero and the associated *srev_flags* field is also zero, the firmware will skip srev checking. Intel shall not sign an NVM image in which either of these fields are set to zero with production keys, and customers are discouraged from doing so if they opt to own the Signature Root of Trust. An image with an srev of zero and *srev_flags* of zero can only be prevented from executing on the device if the ISK that signed it has been revoked.

There are multiple instances of the Security Revision (*lad_srev*) field in an NVM image, which, if the *srev_in_fuses* fuse is blown should use the *srev_flags* field in the CSS headers to map to revision fuses as follows:

- NVM module security revision — Maps to the NVM Bank srev.
- Extended mini-loader security revision — Maps to the NVM Bank srev.
- Recovery firmware security revision — Maps to the NVM Bank srev.
- OROM security revision — Maps to the OROM srev.
- Netlist security revision — Maps to the Netlist srev.

Note: ISKs have no security revision field, and are revoked via in-field fuse programming, rather than via srev increases.

The following policy applies:

- NVM module security revision, Extended Mini-loader and Recovery Firmware security revision shall increase together since all of them are compared against the same fused srev (if the srev_in_fuses fuse is blown).
- OROM security revision can increase independently of others.
- Netlist security revision can increase independently of others.
- The device firmware shall ensure that when srev fuses are programmed in field, bits that were already programmed will not be programmed again.
- Upon any NVM Bank srev updates the firmware shall ensure that the srev programmed is no greater than the srev of the currently valid NVM Bank, and is no less than the srev of the previous image.

Notes:

- Since srev values are (optionally) programmed in fuses, Srev values in the E610 are a monotonically increasing series of bits: 0x0 -> 0x1 -> 0x3 -> 0x7...0xFFFFFFFFFFFFFFFF. For a 64-bit srev field, a total of 65 values (including the special case of zero) are thus possible. Thus, if fuse-based SREVs are enabled, the maximum value is determined by the number of available fuses and the upper bits of the *lad_srev* field in the CSS header may never be set.
- The *lad_srev* in the CSS header is encoded as a standard binary value. Therefore, comparing a fused value to a field in the CSS header requires format conversion.
- Not all NVM updates need to have an incremental security revision. A roll back of updates to non-security related parameters (such as firmware patches where the Security Revision field is equal to the existing image) are allowed.
- The size field gives the entire module size including the CSS header itself. It is expressed in DWords.

The usage of SREV is optional, and users should opt in for the SREV to take effect. Opting in is done by updating the matching Min Srev value in the MinSrev TLV (for NVM bank, OROM) and NetlistMinSrev TLV (for Netlist).

Note: Only the NVM Module and OROM module Min Srev are represented in the MinSrev TLV, as the Recovery Firmware Security Revision is always the same as the NVM module Security Revision, and the Extended Mini-loader Security Revision is always lower or equal to the NVM module Security Revision.

OEMs may choose to use their own signing service rather than an Intel signing service for signing custom OROM and netlist images.

3.3.9.3 Netlist Authentication During Update

The following figure shows the netlist update flow.

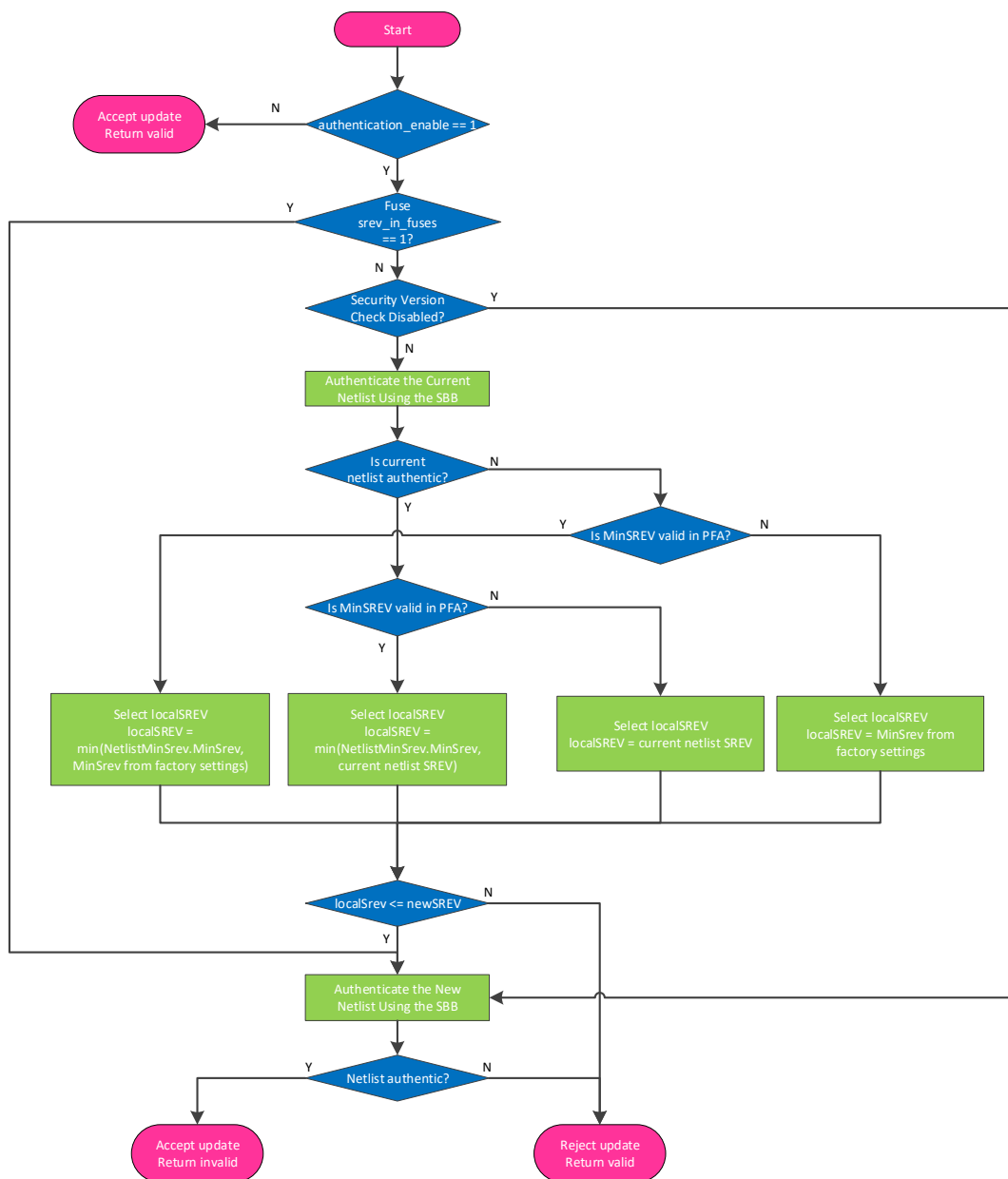


Figure 3-15. Netlist Authenticated Update Flow

The detailed flow is described below:

If authentication is disabled in fuse, the new netlist updated without authentication.

If authentication is enabled and srev_in_fuses fuse is blown, the new netlist will be authenticated by the SBB and accepted if the authentication passes. The SREV check is done by the SBB.

If authentication is enabled and srev_in_fuses fuse is not blown:

- If Security Version Check is disabled, the new netlist will be authenticated and updated if the authentication passes.
- If Security Version Check is enabled:
 - If the netlist in the current active bank (current netlist) is authentic:
 - If NetlistMinSrev in PFA is valid and the new netlist lad_srev is greater than or equal to minimum(NetlistMinSrev.MinSrev, lad_srev of current netlist), the new netlist will be authenticated and updated if the authentication passes
 - If NetlistMinSrev in PFA is not valid and the new netlist lad_srev is greater than or equal to lad_srev of current netlist, the new netlist will be authenticated and updated if the authentication passes
 - If the netlist in the current active bank (current netlist) is not authentic:
 - If NetlistMinSrev in PFA is valid and the new netlist lad_srev is greater than or equal to minimum(NetlistMinSrev.MinSrev, MinSrev from factory settings), the new netlist will be authenticated and updated if the authentication passes

3.3.9.4 Protected Modules

Any data that is modified in the field (either by the OEM during manufacturing or by the end user) cannot be included in the signed region of the NVM. The E610 cannot generate a signed image by itself because the private key is not available to it to generate the digital signature in the NVM.

The following NVM modules may require authentication in E610 based on the fuse configuration. Each module is with its own digital signature:

- Option ROM
- NVM bank (including the mini-loader)
- Netlist

3.3.9.5 Software Requirements

A software tool must prepare NVM images for the CSS signing step, pre-pending the CSS manifest, applying an Intel security revision field. After receiving the signed image, the tool merges the excluded fields back into the NVM image and performs an internal integrity check to verify that the merge was successful (such as a software computation of the digital signature passes).

SVTools (LANConf) MUST implement an NVM update image integrity check option in software prior to applying NVM updates to hardware. SVTools might integrate capabilities to generate self-signed NVM images to assist in the SV and debug process by developers.

CELO (or equivalent manufacturing diagnostics tool) must implement a test to check the GLNVM_FL A Locked bit state as part of manufacturing qualification. If after the NVM is programmed, and the E610 powers up with the GLNVM_FL A.Locked Bit not set, an inappropriate NVM image has been loaded during manufacturing (an NVM image with incorrect Flash opcodes). This represents a critical error.

With GLNVM_FLA unlocked, unauthorized in-the-field updates can bypass designed firmware integrity checks.

Note: On parts with the *bfp_dis* fuse blown, GLNVM_FLA.Locked will be set even if image programming has failed as the device is prevented from entering Blank Flash Programming mode. In this case, the manufacturing diagnostics tool must check for a valid firmware being loaded by configuring an Admin Queue, and issuing the Get Version (opcode 0x0001) AQ command to the firmware to confirm that the correct firmware has been programmed and is functional.

Host software device drivers might implement an interface enabling a network administrator to perform an internal verification check of the signed NVM image. Using Windows drivers, this would take the form of an OID, which reports a SUCCESS or INVALID_PARAMETER. Using Linux, an **ethtool** command extension is advised to enable command line interrogation of the NVM content using the hash value build into the hardware as well as the saved CSS manifest in the NVM image.

3.3.9.6 Manufacturing Requirements

3.3.9.6.1 Factory Settings Preservation Flow

Manufacturing information should be stored at the end of board manufacturing to the "Factory Settings" section that is read only to software and is located outside of NVM banks. This section includes the active PFA, active Topology Netlist and a 32B header.

Update of this section can be done only once, by firmware via AQC "Save Factory Settings" (Opcode 0x708) [Section 3.3.5.8](#) or by software tools in Blank Flash Programming mode.

The AQC will fail with EMODE error if the 'Factory Settings' section is not entirely erased.

This section will never be modified in any NVM update.

The entire active PFA can be reset to factory settings by software via AQC "NVM Write Activate Command (see [Section 3.3.10.8](#)) using preservation mode 10b.

Software default to recover PFA & netlist should be "return to factory settings".

If software uses "return to signed default" option, following the update firmware is required to restore MAC Addresses from factory settings at next firmware load.

After recovery has been completed and normal mode firmware is running, software can do additional updates to modify PFA.

3.3.9.6.2 End-of-Line Verification

OEM's must execute CELO (or an equivalent manufacturing diagnostics tool) to verify the GLNVM_FLA Locked bit state (as previously described).

3.3.10 NVM Access Admin Commands and Events

NVM access commands are not supported when the E610 is in the blank Flash programming mode. They are available only to the PFs once it has acquired NVM ownership via the commands described in [Section 10.11.2.7.3](#).

Table 3-22. NVM Access Admin Commands and Events

Command/Event	Opcode	Brief Description	Detailed Description
NVM Read	0x0701	Read a segment from the NVM into a host buffer.	Section 3.3.10.1
NVM Erase	0x0702	Erase consecutive 4 KB sectors of the Flash.	Section 3.3.10.2
NVM Write	0x0703	Write a segment of the NVM from a host buffer.	Section 3.3.10.3
NVM Config Read	0x0704	Read feature selections.	Section 3.3.10.4
NVM Config Write	0x0705	Program feature selections.	Section 3.3.10.5
NVM Update Checksum	0x0706	Update PFA checksum.	Section 3.3.10.7
NVM Write Activate	0x0707	Activate new NVM.	Section 3.3.10.8
Save Factory Settings	0x0708	Save Factory Settings.	Section 3.3.10.9
NVM Update EMPR	0x0709	Reset after new firmware update.	Section 3.3.10.10
Set Package Data	0x070A	Provide package data to firmware.	Section 3.3.10.11
Pass Component Table	0x070B	Check if new module is compatible with current one.	Section 3.3.10.12
NVM Sanitization	0x070C	supports NIST SP800-88 "Clear" operation	Section 3.3.10.13

Note: All parameters in the admin commands are defined in little endian.

3.3.10.1 NVM Read (0x0701)

Table 3-23. NVM Read Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 10.11.2.4.2.1 for details.
Opcode	2-3	0x0701	Command opcode.
Datalen	4-5		Length in bytes of command buffer
Return Value/VFID	6-7		Return value. Zeroed by the device driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Offset	16-18		Offset in the module, which is expressed in bytes from the pointed module's beginning. This is the byte offset of the first byte returned in the data buffer.
Command Flags	19		NVM Access Admin Command Parameters Bit 0: Last Command Bit Used to notify EMP that this is the last admin command of a sequence. Bit 7: Read from Flash When Bit 7 is set, the read is done directly from the Flash and not from Shadow RAM. Relevant only if access address is below 64 KB. Ignored otherwise, so above 64 KB the read is always from the Flash, only when <i>Module_TypeID</i> is zero. All other bits = Reserved. Must be zeroed. Only one Module type is read at a time. For PFA TLVs only the length+data portion of the TLV is read (not the <i>TLV Type</i>). See Table 6-6 for TypeID values.
Module_TypeID	20-21		A value of 0x0000 means that the command is performed over the Flash part seen as a flat memory.
Length	22-23		Length of the section to be read, which is expressed in bytes from the offset in the module. A value of 0xFFFF means the last byte to be returned is the last byte of the module (if byte 17 was not set to 0x0000). In any case, a (single) read command is limited up to 4 KB.
Reserved	24-31		Reserved.

Table 3-24. NVM Read Response

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 10.11.2.4.2.2 for details.
Opcode	2-3	0x0701	Command opcode.
Datalen	4-5		Number of bytes that were read. If a value of 0xFFFF was set in the admin command (and if byte 17 was not set to 0x0000), this field returns the length from the offset to the module's end unless it is above 4K.
Return Value	6-7		Return value. 0x0 = No error (success). EPERM = The module pointer location specified in the command does not permit the required operation. The word contents is not a pointer. EINVAL = Out of range offset/length (beyond the module's size). EIO = Flash defect. EBUSY = The PF is not permitted to post this command because it does not own the NVM resource. This error code is also returned if the PF attempts to post a command while another NVM command is in process.
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Offset	16-18		Offset in the module, copied from the command.
Reserved	19		Reserved.
Module_TypeID	20-21		<i>Module_TypeID</i> copied from the command.
Length	22-23		Length in bytes of command buffer.
Reserved	24-31		Reserved.

3.3.10.2 NVM Erase (0x0702)

This command is used to erase the contents of 4 KB Flash sectors.

Table 3-25. NVM Erase Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.1.1 for details.
Opcode	2-3	0x0702	Command opcode.
Datalen	4-5		Must be zeroed by the software device driver.
Return Value/VFID	6-7		Must be zeroed by the software device driver.
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Reserved	16-18		Reserved.
Command Flags	19		NVM Access Admin Command Parameters Bit 0: Last Command Bit Used to notify the EMP that this is the last admin command of a sequence. All other bits = Reserved. Must be zeroed.
Module_TypeID	20-21		The only valid TypeIDs are: 1st NVM Bank Pointer (0x42): Non-valid bank erase. 1st OROM Bank Pointer (0x44): Non-valid bank erase. 1st TLV Extension Bank Pointer (0x46): Non-valid bank erase. - One of the scratch pad pointers (0x4B, 0x50, 0x59).
Reserved	22-23		Reserved.
Reserved	24-31		Reserved.

This is an asynchronous command. The EMP reads the command from the Tx descriptor and writes back an immediate completion, intended only as an ACK/NACK that the command has been addressed by the EMP. The EMP checks the validity of the command and returns an error (NACK) on the Tx descriptor completion if it is unable to process the command. If successful (ACK), the EMP then schedules the NVM Erase operation to be performed by a lower priority thread, which can be preempted by other AQ commands.

Once completed, the EMP posts a completion event on the Rx descriptor. Software must hold the NVM resource lock while performing this operation and must release it once the full NVM operations complete (assuming the erase is part of an update sequence).

Software must not post another NVM command while this command is in process. For example, during the time between posting the request on the Tx descriptor and receiving the completion event on the Rx descriptor.

Table 3-26. NVM Erase Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.1.2 for details.
Opcode	2-3	0x0702	Command opcode.
Datalen	4-5		Reserved.
Return Value	6-7		Return value. 0x0 = No error (success). EPMR = The module pointer location specified in the command does not permit the required operation. The word contents is not a pointer to a next bank area. EBUSY = The PF is not permitted to post this command because it does not own the NVM resource. This error code is also returned if the PF attempts to post a command while another NVM command is in process. EACCESS = Attempt to erase a non erasable area.
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Reserved	16-31		Reserved.

Table 3-27. NVM Erase Completion

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.1.2 for details.
Opcode	2-3	0x0702	Command opcode.
Datalen	4-5		Reserved
Return Value	6-7		Return value. 0x0 = No error (success).
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Reserved	16-17	0x0	Reserved. Must be zeroed
Reserved	19		Reserved.
Module_TypeID	20-21		Copied from the command.
Length	22-23		Copied from the command. If a value of 0xFFFF was set in the admin command, this field returns the length from the offset to the module's end in 4 KB units.
Reserved	24-31		Reserved.

3.3.10.3 NVM Write (0x0703)

This command is used to write the data given by the attached buffer into a specified location in the NVM. Erasing the relevant sector(s) by posting NVM erase command(s) (see [Section 3.3.10.2](#)) is required prior to posting this command.

For double bank updates (like Module_TypeId = 0x42, 0x44, 0x46), the command must be followed by the NVM Write Activate command for a bank update to take effect.

Table 3-28. NVM Write Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.2.1 for details.
Opcode	2-3	0x0703	Command opcode.
Datalen	4-5		Length in bytes of command buffer.
Return Value/VFID	6-7		Must be zeroed by the software device driver.
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Offset	16-18		Offset, which is expressed in bytes from the pointed module's beginning. This is the byte offset of the first byte to be written. Offset of zero points to the section length. However the length is not overwritten.
Command Flags	19		NVM Access Admin Command Parameters Bit 0: Last Command Bit Used to notify EMP that this is the last admin command of a sequence. Note: Setting the <i>Last</i> bit causes a Shadow RAM dump, but it is recommended to use the NVM Write Activate to achieve this. Bit 6: Allow special update (set to allow change of MinSrev TLV). All other bits = Reserved. Must be zeroed.

Table 3-28. NVM Write Command [continued]

Name	Byte.Bit	Value	Remarks
Module_TypeID	20-21		Module TypeID The only valid init module pointers are: • 0x00: Flat memory view • 1st NVM bank pointer (0x42): Non-valid bank. • 1st OROM bank pointer (0x44): Non-Valid Bank. • 1st TLV Extension Bank Pointer (0x46): Non-valid bank. • One of the scratch pad pointers: — Firmware scratch pad pointer (0x50) — Link Topology scratch pad pointer (0x4B) • PFA TypeIDs not included in the list below (see Table 6-6, "Module TypeID Table"). A value of 0x00 here means that the command is performed over the Shadow RAM part seen as a flat memory. No reset or pointer switch is initiated by the E610 in such a case. A flat write attempt to the first 64 KB of the Flash is performed against the Shadow RAM first, and then dumped to the next Shadow RAM bank in the Flash (see Section 3.3.5.4). A flat write (0x00) attempt to a Shadow RAM area other than within the PFA, or words listed below or out side the Shadow RAM is rejected as well. The following PFA TLV sections cannot be updated by this AQC: • Immediate values • Feature configuration • POR auto-load section • PCIR auto-load section • TLVs Header • VPD when the VPD_WRITE_ENABLE is cleared • HII port/PF disable • Padding TLV • Software Checksum The MinSrev and NetlistMinSrev modules can be updated only if <i>Allow Special Update</i> is set. The following checks are applied when the MinSrev and NetlistMinSrev modules are updated: • MinSrev valid is not cleared (1 → 0). • If MinSrev valid and Current MinSrev ≤ Srev in signed image, then check that Current MinSrev < New Srev ≤ Srev in signed image. • If Current MinSrev valid and Current MinSrev > Srev in signed image OR Current MinSrev NOT valid, then check that New NVM Srev ≤ NVM Srev in signed image. Note: Srev in signed image = Srev of the new image if exists - if not of current image. Note: MinSrev and NetlistMinSrev cannot be updated in recovery mode. In debug mode, a value of 0x00 writes to the entire Flash seen as flat memory without any protection. Only one module is updated at a time. For TLVs only the data portion of the TLV is updated.
Length	22-23		Length of the section to be written, which is expressed in bytes from the offset in the module. A (single) write command is limited up to 4 KB and must not spread over two (consecutive) 4 KB sectors. Also, attempting to write a RO word invalidates the entire command.Length is truncated to size of TLV in bytes +2.
Reserved	24-31		Reserved.

This is an asynchronous command. EMP reads the command from the Tx descriptor and writes back an immediate completion, intended only as an ACK/NACK that the command has been addressed by the EMP. The EMP checks the validity of the command and returns an error (NACK) on the Tx descriptor completion if it is unable to process the command. If successful (ACK), the EMP then schedules the NVM write operation to be performed by a lower priority thread, which can be preempted by other AQ commands.

Once completed, the EMP posts a completion event on the Rx descriptor. Software must hold the NVM resource lock while performing this operation and must release it once NVM operations complete.

Software must not post another NVM command while this command is in process. For example, during the time between posting the request on the Tx descriptor and receiving the completion event on the Rx descriptor.

Table 3-29. NVM Write Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.2.2 for details.
Opcode	2-3	0x0703	Command opcode.
Datalen	4-5		Length in bytes of command buffer.
Return Value	6-7		Return value. 0x0 = No error (success). EPERM = The module type specified in the command does not permit the required operation. The word contents is not a pointer, or an attempt to write a RO module or word, or command is received not at the right place in the flow. EINVAL = Out of range offset/length (beyond the relative free area module's limits), or write spread over two (consecutive) sectors. EBUSY = The PF is not permitted to post this command because it does not own the NVM resource. This error code is also returned if the PF attempts to post a command while another NVM command is in process. EACCESS = Attempt to write to read only area or authentication failure.
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Reserved	16-31		Reserved.

Table 3-30. NVM Write Completion

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.2.2 for details.
Opcode	2-3	0x0703	Command opcode.
Datalen	4-5		Length in bytes of command buffer.
Return Value	6-7		Return value. 0x0 = No error (success). EIO = Flash defect. EACCES = Security check failed: • Public key check failed. • Module digest (CRC) check failed. • Module security revision check failed. • Device ID check failed. • Module ID check failed. EINVAL = Invalid parameters
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Offset	16-18		Copied from the command.
Response Flags	19		RESET_FLAG Bit 1:0: A reset flag to indicate the type of reset required to get the NVM bank update effective. 00b = POR 01b = PERST 10b = EMPR All other bits = Reserved Note: Relevant only for NVM Bank update.
Module_TypeID	20-21		Module_TypeID that was used for this update
Length	22-23		Copied from the command.
Data Address High	24-27		Address of command buffer.
Data Address Low	28-31		

3.3.10.4 NVM Config Read (0x0704)

This admin command reads currently configured feature/super-feature selections and immediate field values/entries. The features/fields to be read are specified in the command's buffer. It can also be used for reading all features or fields. In case one buffer is not enough Feature or field iteration is used. The next *Feature_ID/Field_ID* to read are returned in the command response in this case.

Table 3-31. NVM Config Read Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.2.1 for details.
Opcode	2-3	0x0704	Command opcode.
Datalen	4-5		Length in bytes of command buffer.
Return Value/VFID	6-7		Must be zeroed by the software device driver.
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Command Flags	16		NVM access admin command parameters. Bit 0: Single/Multiple elements. 0b = Only a single <i>Feature_ID/Field_ID</i> is read. 1b = <i>Feature_ID/Field_ID</i> iteration is used. Bit 1: Feature/Field 0b = Feature selections are read. 1b = Immediate fields are read. All other bits = Reserved. Must be set to zero.
Reserved	17	0x0	Reserved. Must be set to zero.
Element Count	18-19	0x0	The number of features/fields returned. Zeroed by driver, written by firmware.
Feature_ID/ Field_ID	20-21	See description	Single <i>Feature_ID/Field_ID</i> when Bit 16.0 is 0b. <i>Feature_ID</i> when Bit 16.1 is 0b. <i>Field_ID</i> when Bit 16.1 is 1b. Iterator: <i>Feature_ID/Field_ID</i> to start reading from (iterator) when Bit 16.0 is 1b. Note: When Bit 16.0 is 0b, <i>Feature_ID/Field_ID</i> = 0 is not valid. When Bit 16.0 is 1b, it indicates that the command reads the data starting from the first <i>Feature_ID/Field_ID</i> in the array.
Reserved	22-31		Reserved.

Table 3-32. NVM Config Read Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.2.2 for details.
Opcode	2-3	0x0704	Command opcode.
Datalen	4-5		Length in bytes of command buffer.
Return Value	6-7		Return value. 0x0 = No error (success).
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Command Flags	16		NVM access admin command parameters. Bit 0: Single/Multiple elements. 0b = Only a single <i>Feature_ID/Field_ID</i> is read. 1b = <i>Feature_ID/Field_ID</i> iteration is used. Bit 1: Feature/Field 0b = Feature selections are read. 1b = Immediate fields are read. All other bits = Reserved. Must be set to zero.
Reserved	17	0x0	Reserved. Must be set to zero.
Element Count	18-19	See description	The number of features/fields returned.
Feature_ID/Field_ID	20-21	See description	Same as the command.
Reserved	22-31		Reserved.

3.3.10.5 NVM Config Write (0x0705)

This admin command writes the feature selections and the values of the immediate fields provided in the attached command buffer to the NVM.

After this command is completed, an [NVM Write Activate \(0x0707\)](#) must be called to make the configuration persistent.

Table 3-33. NVM Config Write Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.2.1 for details.
Opcode	2-3	0x0705	Command opcode.
Datalen	4-5		Length in bytes of command buffer.
Return Value/VFID	6-7	0x0	Must be zeroed by the software device driver.
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Command Flags	16		NVM Access Admin Command Parameters Bit 0: Reserved. Bit 1: Feature/Field 0b = Feature selections are written. 1b = Immediate fields are written. Bit 2: Add New Configuration 0b = Existing configuration. 1b = New configuration added. All other bits = Reserved. Must be set to zero.
Reserved	17	0x0	Reserved. Must be set to zero.
Element Count	18-19	0x0	The number of features/fields in the command buffer.
Reserved	20-23	0x0	Reserved. Must be set to zero.
Reserved	24-31		Reserved.

Table 3-34. NVM Config Write Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.2.2 for details.
Opcode	2-3	0x0705	Command opcode.
Datalen	4-5		Length in bytes of command buffer.
Return Value	6-7		Return value. 0x0 = No error (success).
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Reserved	16-31		Reserved.

3.3.10.6 NVM Config Read/Write Command Buffer

Described below is the format of the command buffer attached to the NVM Config Read Response and to the NVM Config Write commands. The buffer can either be filled with Feature Selection fields or Immediate Field fields, depending on Bit 16.1 in the command.

Table 3-35. Feature Buffer for NVM Config Read/Write

Parameter	Bytes.Bits	Description
Feature_ID	0-1	Feature_ID
Feature Flags	2-3	Feature options for NVM_Config_Read only. Reserved for NVM_Config_Write. Bit 0: OEM only (is set). Bits 1:2: Reserved. Bit 3: If set the Feature fields are mapped in DWORD-wise, otherwise are WORD. Bit 4: Should be set when using a POR CSR. Bit 5:15: Reserved.
Feature selection/Field Value	4-5	Configured feature selection for NVM Config Read. Requested feature selection for NVM Config Write.

Table 3-36. Immediate Buffer for NVM Config Read/Write

Parameter	Bytes.Bits	Description
Field_ID	0-1	Field_ID
Field flags	2-3	Field options for NVM Config Read. Reserved for NVM Config Write.
Field Value	4-5	Field value.

3.3.10.7 NVM Update Checksum (0x0706)

This admin command recalculates/verifies the PFA checksum. This command is used by software once it fixed a corrupted PFA or to check the validity of the PFA after reset.

Table 3-37. NVM Update Checksum Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.1.1 for details.
Opcode	2-3	0x0706	Command opcode.
Datalen	4-5	0x0	Length in bytes of command buffer.
Return Value/VFID	6-7	0x0	Must be zeroed by the software device driver.
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Command Flags	16		Bit 0: Verify Checksum Bit 1: Recalculate Checksum All other bits = Reserved. Note: Only one bit of the above can be set in a command. Note: After a recalculate action, the software must initiate a Shadow RAM dump.
Reserved	17-31	0x0	Reserved. Must be set to zero.

Table 3-38. NVM Update Checksum Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.1.2 for details.
Opcode	2-3	0x0706	Command opcode.
Datalen	4-5		Length in bytes of command buffer.
Return Value	6-7	0x0	Return value. 0x0 = No error (success). INVAL = Both 17.0 and 17.1 bits are set.
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Reserved	16-17		Reserved.
Checksum	18-19	0x0	A value of 0xBABA indicates a valid checksum. Returned only if <i>Verify Checksum</i> flag was set in command.
Reserved	20-31		Reserved.

3.3.10.8 NVM Write Activate (0x0707)

This admin command updates the control word with the required banks' validity bits and dumps the Shadow RAM to Flash.

It must be called after NVM Write AQ Command was successfully completed in order for the new bank(s) to be updated.

If no flag is set, this command acts as an SR-dump command.

Table 3-39. NVM Write Activate Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.1.1 for details.
Opcode	2-3	0x0707	Command opcode.
Datalen	4-5	0x0	Length in bytes of command buffer.
Return Value/VFID	6-7	0x0	Must be zeroed by the software device driver.
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Reserved	16-17	0x0	Reserved. Must be set to zero.
Command Flags 2	18	0x0	Bit 0: Set to Elevate next PERST to EMPR. Clear has no effect. All other bits = Reserved. Must be set to zero.
Command Flags	19	0x0	Bit 0: Reserved. Bits 2-1: Preservation Mode (relevant only if Bit 3 is set): 00b = No preservation — Notify EMP to avoid any field preservation during NVM bank update. Returns to the values in the copy of the PFA stored in the signed NVM module. 01b = Preserve all (Default) — Used to notify EMP to preserve all the necessary customization fields during NVM bank update. 10b = Return to factory settings. 11b = Preserve Only Selected Fields — Used to notify EMP to preserve only the immediate fields and section marked as such during NVM bank update. The following sections are currently included: • PCI Serial ID • PF MAC Addresses TLV • MNG MAC Address TLV section • HLP MAC Address TLV section • SMBus addresses • MinSrev • NetlistMinSrev Bit 3: 0b = Keep current NVM Bank. 1b = Switch to invalid NVM Bank Bit 4: 0b = Keep current NV OROM Bank. 1b = Switch to invalid OROM Bank Bit 5: 0b = Keep current EXT TLV Bank. 1b = Switch to invalid EXT TLV Bank Bit 6: 0b = No event. 1b = Revert Last Activate. Bit 7: Reserved. Note: If Bits 3:5 are clear, Shadow RAM is dumped and swapped with no validity bits change.
Reserved	20-31	0x0	Reserved. Must be set to zero.

Table 3-40. NVM Write Activate Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.1.2 for details.
Opcode	2-3	0x0707	Command opcode.
Datalen	4-5		Length in bytes of command buffer.
Return Value	6-7		Return value. 0x0 = No error (success). EPMR = Returned when at least one of the specified bank(s) is not in the expected state for being updated, or command is received not at the right place in the flow. EMODE =NVM module not included in activation.
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Reserved	16-31		Reserved.

Table 3-41. NVM Write Activate Completion

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.1.2 for details.
Opcode	2-3	0x0707	Command opcode.
Datalen	4-5		Length in bytes of command buffer.
Return Value	6-7		Return value. 0x0 = No error (success). 0xC = EBUSY - Unavailable resource. 0xA = EACCES - Not available for activation (no image was loaded successfully). 0x15 = EMODE - Internal firmware error.
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Reserved	16-31	0x0	Reserved.

3.3.10.9 Save Factory Settings (0x0708)

This admin command saves the PFA, active Topology Netlist, and 32B header to a permanent read-only NVM location. This command should be executed at the end of product manufacturing after NVM content has been programmed, including MAC Addresses. For more information refer to [Section 3.3.9.6.1, "Factory Settings Preservation Flow"](#).

Table 3-42. Save Factory Settings Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.1.1 for details.
Opcode	2-3	0x0708	Command opcode.
Datalen	4-5	0x0	Length in bytes of command buffer.
Return Value/VFID	6-7	0x0	Must be zeroed by the software device driver.
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Reserved	16-31	0x0	Reserved. Must be set to zero.

Table 3-43. Save Factory Settings Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.1.2 for details.
Opcode	2-3	0x0707	Command opcode.
Datalen	4-5	0x0	Length in bytes of command buffer.
Return Value	6-7	0x0	Return value. 0x0 = No error (success). EACCESS = Save Factory Settings command was previously executed.
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Reserved	16-31	0x0	Reserved.

3.3.10.10 NVM Update EMPR (0x0709)

This admin command allows the software the request an EMPR after a successful reset to allow activation of the new firmware.

Table 3-44. NVM Update EMPR Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.1.1 for details.
Opcode	2-3	0x0709	Command opcode.
Datalen	4-5	0x0	
Return Value/VFID	6-7	0x0	Must be zeroed by the software device driver.
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Reserved	16-31	0x0	Reserved. Must be set to zero.

Table 3-45. NVM Update EMPR Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.1.2 for details.
Opcode	2-3	0x0709	Command opcode.
Datalen	4-5		Length in bytes of command buffer.
Return Value/VFID	6-7		Return value. 0x0 = No error (success). EMODE = No successful update pending. Reset not allowed. EPRM = Command is received not at the right place in the flow.
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Reserved	16-31	0x0	Reserved.

3.3.10.11 Set Package Data (0x70A)

This command is equivalent to the reception of a PLDM Firmware Update *GetPackageData* command. This command assumes that any package data is smaller than 4K and hence is contained within a single command. This command should be sent as part of the NVM update as the first command in the flow.

Table 3-46. NVM Set Package Data Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.1.1 for details.
Opcode	2-3	0x070A	Command opcode.
Datalen	4-5		Length in bytes of command buffer.
Return Value/VFID	6-7		Must be zeroed by the software device driver.
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Reserved	16-18		Reserved.
Command Flags	19		Bit 0: Delete PackageData If set, the current PackageData stored by firmware is deleted. If this bit is set, buffer size is zero. All other bits = Reserved. Must be set to zero.
Reserved	20-31		Reserved.

Table 3-47. NVM Set Package Data Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.1.2 for details.
Opcode	2-3	0x070A	Command opcode.
Datalen	4-5		
Return Value	6-7		Return Value. 0x0 = No error (success). EPerm = NVM update is not allowed (between activate and reset or lock-down). EINVAL = Error in parsing of the content of the package data. EBUSY = The PF is not permitted to post this command because it does not own the NVM resource. This error code is also returned if the PF attempts to post a command while another NVM command is in process. EACCESS = Attempt to request an update of a RO area.
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Reserved	16-18		Reserved.
Flags	19		Bit 0: Skipped Entries If set, some of the update requests will be skipped due to unknown PFA TLVs or unknown VPD keys. All other bits = Reserved.
Reserved	20-31		Reserved.

3.3.10.11.1 Buffer for NVM Set Package Data Command

The content of the buffer is the *PackageData* as read by the tool from the PLDM header. The size of this buffer is up to 1 KB.

The content is as follows:

- Byte [0].Bit 0 — PFA Preserve (default mode). Should be set to zero.
 - Byte [0].Bit 1: Reserved
 - Byte [0].Bits [7:2] — Reserved
 - Byte[1] — Reserved
 - TLV as follows:
 - Type — 2 bytes:
 - GFID - 0x1
 - Length — 2 bytes:
 - 36 words
 - Value:
 - 17:0 = Current GFID
 - 35:18 = Original GFID
 - The following set of TLVs as needed:
 - Type — 2 bytes:
 - TLV + Offset update - 0x10 // used to update TLVs
 - Length — 2 bytes:
 - According to size of value (in words)
 - Value:
 - uint16: TLV number
 - uint16: Offset in words (offset 0 is the beginning of the TLV data)
 - uint16: Data length in words
 - Variable: Data
 - The following set of TLVs as needed:
 - Type — 2 bytes:
 - VPD Key update - 0x11 // Used to update VPD entries
 - Length — 2 bytes:
 - According to size of value (in words)
 - Value:
 - uint16: VPD key (ascii - e.g. V2 = 0x5632 (ascii('V'), ascii('2')))
 - uint16: Data length (in bytes) // should match current VPD entry size - error otherwise
- Note:** If Data Length is odd, add a pad byte at end of data to make the TLV word aligned.

- Variable: Data

Notes:

- The TLVs can be in any order.
- Although this command is received at the beginning of the update flow, the actual programming of the PFA happens at the end (before the activation).
- The treatment of a PackageData received via *GetPackageData* PLDM firmware update command should be the same as described above.
- For all changes received via this package, regular PFA protection applies as defined in NVM Write admin command. The MinSrev and NetlistMinSrev can be changed according to the regular checks.
- After updating the VPD, the checksum in the VPD RV tag should be updated.
- If a modification of an unknown PFA tag or VPD key is requested, the command is accepted, and this tag/key is ignored. This is indicated via the *Skipped Entries* flag.
- To be able to modify parts of the VPD RO tag, the *VPD Write Enable* bit in NVM should be set. If not set, and modifications of the RO VPD area are requested the command will fail.
- If command 0x070A is received more than once, the last version is retained, even if faulty.
- If unknown TLVs or PFA TLVs with wrong size or VPD keys with wrong size are part of the PackageData, the command will fail.
- An NVM Write Activate command (0x0707) or PLDM *CancelUpdate* or PLDM *ActivateFirmware* or PLDM timeout will invalidate the content received in this command. For a new update, the command should be sent again.

Note: In Recovery Firmware mode, the following applies for the PackageData:

- The flags must follow the same rules as the normal firmware.
- The GFID check is not done, as the current GFID is not trusted.
- TLV data is ignored and no changes are done to the PFA. The *Skipped Entries* flag is set in the response. This also means that an update in Recovery mode that successfully restored to regular firmware should be reapplied when the regular firmware is running, to apply PFA change.

3.3.10.12 Pass Component Table (0x70B)

This command is equivalent to the reception of a PLDM firmware Update *PassComponentTable* command. This command is sent once per component. This command should be sent as part of the NVM update only after the Set Package Data command is sent and before the update actually starts. The E610 assumes these commands are going to be sent until the *TransferFlag* is set to *End* or *StartAndEnd*.

Table 3-48. NVM Pass Component Table Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.2.1 for details.
Opcode	2-3	0x070B	Command opcode.
Datalen	4-5		Length in bytes of command buffer.
Return Value/VFID	6-7		Must be zeroed by the software device driver.
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Reserved	16-18		Reserved.
TransferFlag	19		Possible values: 0x1 = Start 0x2 = Middle 0x4 = End 0x5 = StartAndEnd Note: This <i>TransferFlag</i> is ignored by firmware for now.
Reserved	20-31		Reserved.

Table 3-49. NVM Pass Component Table Response

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.2.2 for details.
Opcode	2-3	0x070B	Command opcode.
Datalen	4-5		
Return Value	6-7		Return Value. 0x0 = No error (success). EPMR = NVM update is not allowed (between activate and reset or lock-down). EBUSY = The PF is not permitted to post this command because it does not own the NVM resource. This error code is also returned if the PF attempts to post a command while another NVM command is in process. EINVAL = Wrong parameters.
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
ComponentResponse	16		<i>ComponentResponse</i> as described in DSP0267: 0 = Component can be updated - <i>ComponentResponseCode</i> must be set to 0x00. 1 = Component cannot be updated - A <i>ComponentResponseCode</i> greater than zero must be provided to explain the reason why the component cannot be updated, or if a flag is required to be set in <i>UpdateOptionFlags</i> field within the <i>UpdateComponent</i> request. 2 = Component should not be updated with this image (this is not part of the DSP0267 spec, but provides more information to host). The matching <i>ComponentResponseCodes</i> are denoted below. All other values reserved. This value reflects the update status of this module. Possible <i>ComponentResponse</i> in Recovery Firmware: Netlist: Same as in regular firmware specified above (0 1 2). NVM: 1 = Component should not be updated with this image in case of <i>ComponentResponseCode</i> 0x6 or 0x5 or 0x7. 2 = Partial Check.

Table 3-49. NVM Pass Component Table Response [continued]

Name	Byte.Bit	Value	Remarks
ComponentResponseCode	17		<i>ComponentResponseCode</i> as defined in DSP0267: 0x00 = Component can be updated. 0x01 = Component comparison stamp is identical to the firmware component comparison stamp in the FD. The device accepts such an update, but the software should reflect this to the user. 0x02 = Component comparison stamp is lower than the firmware component comparison stamp in the FD (downgrade). The device accepts such an update, but the software should reflect this to the user. 0x03 = Invalid component comparison stamp. The device rejects the request and returns a <i>ComponentResponse</i> = 2. Returned, among others, if GFID does not match. 0x04 = Component has conflict with another component provided in a separate <i>PassComponentTable</i> command. 0x05 = Prerequisites for this component have not been met. Returned if 0x70A was not sent before this command. The device rejects the request and returns a <i>ComponentResponse</i> = 2. 0x06 = Component is not supported on FD. The device rejects the request and returns a <i>ComponentResponse</i> = 2. 0x07 = Security restrictions prevent component from being downgraded. Should include Srev check. The device rejects the request and returns a <i>ComponentResponse</i> = 2. 0x08 = Incomplete component image set was received. Not used. 0x09 = Reserved. 0x0A = Component version string is identical to the firmware component version string in the FD. The device accepts such an update, but the software should reflect this to the user. 0x0B = Component version string is lower to the firmware component version string in the FD. The device accepts such an update, but the software should reflect this to the user. All other values are reserved. In case of Recovery Firmware, the following values are returned based on partial data: 0x00 = Component can be updated. 0x05 = Pre-requisites for this component have not been met. Returned if 0x70A was not sent before this command. The device rejects the request and returns a <i>ComponentResponse</i> = 2 0x06 = Component is not supported on FD. The device rejects the request and returns a <i>ComponentResponse</i> = 2 0x07 = Security restrictions prevent component from being downgraded. Should include Srev check. The device rejects the request and returns a <i>ComponentResponse</i> = 2 (NVM SREV) 0x0A = Component version string is identical to the firmware component version string in the FD. The device accepts such an update, but the software can reflect this to the user. (netlist) 0x0B = Component version string is lower to the firmware component version string in the FD. The device accepts such an update, but the software can reflect this to the user. (netlist)
Reserved	18-31		Reserved.

3.3.10.12.1 Buffer of NVM Pass Component Table Command

The buffer of NVM Pass Component Table command contains the Component Table as passed to the device by the PassComponentTable PLDM firmware command as described in DSP0267 *PassComponentTable* command and described in [Table 3-50](#).

Table 3-50. Component

Type	Name	Format/Value
uint16	ComponentClassification	Should be 0x000A - Firmware for all components.
uint16	ComponentIdentifier	NVM = 0x6 OROM = 0x5 Topology Netlist = 0x8
uint8	ComponentClassificationIndex	0x0 - Not used.
uint32	ComponentComparisonStamp	As defined in Section 10.8.5.2.1
enum8	ComponentVersionStringType	ASCII = 1
uint8	ComponentClassificationIndex	The length, in bytes, of the ComponentVersionString.
Variable	ComponentVersionString	As defined in Section 10.8.5.2.1 .

3.3.10.13 NVM Sanitization (0x70C)

This command supports NIST SP800-88 “Clear” operation. It erases all customer data from the device. The following actions are performed:

1. Restore the PFA contents to factory defaults.
2. Clear all VPD data.

Table 3-51. NVM Sanitization Command

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.2.1 for details.
Opcode	2-3	0x070C	Command opcode.
Datalen	4-5		Length in bytes of command buffer.
Return Value/VFID	6-7		Must be zeroed by the software device driver.
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Command Flags	16		Bit 0: REQ Type 0b = Read 1b = Sanitize Bit 1: If REQ Type = Read 0b = Read Sanitization permission (from “EMP Settings” section of the NVM) 1b = Read Sanitization status If REQ Type = Sanitize 0b = Clear operation 1b = Reserved (for Purge operation) All other bits = Reserved. Must be set to zero.
Reserved	17-31		Reserved.

Table 3-52. NVM Sanitization (Immediate) Response (Sent on TXQ)

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.2.1 for details.
Opcode	2-3	0x070C	Command opcode.
Datalen	4-5		Length in bytes of command buffer.
Return Value/VFID	6-7		Return Value 0x0 = No Error (success) 0x12 = EBUSY, applied in case of read /write access clash or Fuses failed. Other values indicate firmware internal errors.
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Command Flags	16		Bit 0: REQ Type 0b = Read 1b = Sanitize Bit 1: If REQ Type = Read 0b = Read Sanitization permission (from "EMP Settings" section of the NVM) 1b = Read Sanitization status If REQ Type = Sanitize 0b = Clear operation 1b = Reserved (for Purge operation) All other bits = Reserved. Must be set to zero.
Values	17		For REQ Type = Read Sanitization permissions: Bit 0: Host can apply Clear NVM operation 0b = no 1b = yes Bit 1: Reserved Bit 2: BMC can apply Clear NVM operation 0b = no 1b = yes Bits 3-7: Reserved For REQ Type = Read Sanitization status: Bit 0: Host applied Clear NVM operation 0b = no 1b = yes Bit 1: Host applied Clear NVM operation status 0b = failed 1b = succeeded Bit 2: BMC applied Clear NVM operation 0b = no 1b = yes Bit 3: Host applied Clear NVM operation status 0b = failed 1b = succeeded Bits 4-7: Reserved For REQ Type = Sanitize, all bits are cleared
Reserved	18-31		Reserved.

Table 3-53. NVM Sanitization Completion (sent on RXQ)

Name	Byte.Bit	Value	Remarks
Flags	0-1	0	See Section 10.11.2.4.2.1 for details.
Opcode	2-3	0x070C	Command opcode.
Datalen	4-5		Length in bytes of command buffer.
Return Value/VFID	6-7		Return Value 0x0 = No Error (success) Other values indicate firmware internal errors.
Cookie High	8-11	Cookie	Opaque value copied by the EMP into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by the EMP into the completion of this command.
Command Flags	16		Bit 0: REQ Type 0b = Read 1b = Sanitize Bit 1: If REQ Type = Read 0b = Read Sanitization permission (from "EMP Settings" section of the NVM) 1b = Read Sanitization status If REQ Type = Sanitize 0b = Clear operation 1b = Reserved for Purge operation All other bits = Reserved. Must be set to zero.
Values	17		For REQ Type = Read Sanitization permissions: Bit 0: Host can apply Clear NVM operation 0b = no 1b = yes Bit 1: Reserved Bit 2: BMC can apply Clear NVM operation 0b = no 1b = yes Bits 3-7: Reserved For REQ Type = Read Sanitization status: Bit 0: Host applied Clear NVM operation 0b = no 1b = yes Bit 1: Host applied Clear NVM operation status 0b = failed 1b = succeeded Bit 2: BMC applied Clear NVM operation 0b = no 1b = yes Bit 3: Host applied Clear NVM operation status 0b = failed 1b = succeeded Bits 4-7: Reserved For REQ Type = Sanitize, all bits are cleared
Reserved	18-31		Reserved.

3.3.11 VPD Support

The Flash image can contain an area for VPD. This area is managed by the OEM vendor and does not influence the behavior of hardware. The VPD area is stored in the PFA with a TLV type of 0x2F. A value of 0b in the GLPCI_CAPCTRL.VPD_EN register bit means VPD is not supported and the VPD capability does not appear in the configuration space. The register bit must be set to 1b in NVM only once the VPD area has been programmed. Refer to [Section 3.3.5.3.1](#).

The maximal VPD area size provisioned in shadow RAM is 1 KB but it can be smaller. The VPD block is built from a list of resources. A resource can be either large or small. The structure of these resources are listed in the following tables.

Table 3-54. Small Resource Structure

Offset	Content
0	Tag = 0xxx,xyyyb (Type = Small(0), Item Name = xxxx, length = yy bytes)
1-n	Data

Table 3-55. Large Resource Structure

Offset	Content
0	Tag = 1xxx,xxxxb (Type = Large(1), Item Name = xxxxxxx)
1-2	Length
3-n	Data

The E610 parses the VPD structure during the autoload process following PCIe reset to detect the read only and read/write area boundaries. The E610 assumes the following VPD fields with the limitations listed:

Table 3-56. VPD Structure Tags

Tag	Length (Bytes)	Data	Resource Description
0x82	Length of identifier string	Identifier	Identifier string.
0x90	Length of RO area	RO data	VPD-R list containing one or more VPD keywords.
0x91	Length of RW area	RW data	VPD-W list containing one or more VPD keywords. This part is optional.
0x78	N/A	N/A	End tag.

VPD structure limitations:

- The structure must start with a tag = 0x82.
- The structure must end with a tag = 0x78 before the shadow RAM's end. The tag must also be word aligned.
- If the E610 does not detect a value of 0x82 in the first byte of the VPD area, or if no end tag is detected, or if the structure does not follow the information listed in Table 3-231, it assumes the area is not programmed:
 - Any read/write access through the VPD registers set are ignored.
 - EMP considers the VPD area as an empty module and thus allows NVM Update commands to be performed over the area pointed by the VPD area pointer, up to the start of another RO module.
 - The VPD pointer itself remains RO.

- The RO area and RW area are both optional and can appear in any order. A single area is supported by per-tag type. Refer to Appendix I in the PCI 3.0 specification for details of the different tags.
- If a VPD-W tag is found, the area defined by its size is writable via the VPD structure.
- The VPD area can be accessed through the PCIe configuration space VPD capability structure listed in [Table 3-56](#). Write accesses to a RO area or any accesses outside of the VPD area via this structure are ignored. If the VPD Write Enable field is set to 1b in the NVM Security Control word, the entire VPD area can be modified via the NVM Update AQ command. Otherwise, the command is completed with an error status.
- VPD area must be mapped into the first valid basic bank of the Flash.
- VPD software does not check the NVM ownership before attempting to access the Flash via dedicated VPD registers (refer to [Section 9.2.3](#)). VPD software write access is recorded in the Flash immediately once the Flash part is available (such as not busy by a previous sector erase operation). Refer to [Section 3.3.5.3](#) for more details.

3.4 Configurable I/O Pins — Software-Definable Pins (SDPs)

The E610 core has four software-defined pins (SDP pins) per port that can be used for miscellaneous hardware or software-controllable purposes. Unless specified otherwise, these pins and their function are bound to a specific LAN device. The use, direction, and values of SDP pins are controlled and accessed by the Extended SDP Control (ESDP) register. To avoid signal contention, following power-up, all four pins are defined as input pins.

Note: SDP direction is controlled by firmware only to avoid electrical contention because of malicious software.

Some SDP pins have specific functionality:

- The default direction of the SDP pins is loaded from the *SDP Control* word in the NVM.
- The lower SDP pins (SDP0-SDP2) can also be configured for use as External Interrupt Sources (GPI). To act as GPI pins, the desired pins must be configured as inputs and enabled by the GPIE register. When enabled, an interrupt is asserted following a rising-edge detection of the input pin (rising-edge detection occurs by comparing values sampled at the internal clock rate, as opposed to an edge-detection circuit). When detected, a corresponding GPI interrupt is indicated in the EICR register.

Note: An SDP configured as output can also generate interrupts, but this is not a recommended configuration.

The bit mappings are shown in [Table 3-57](#) for clarity.

Table 3-57. GPI to SDP Bit Mappings

SDP Pin to be Used as GPI	ESDP Field Settings		Resulting EICR Bit (GPI)
	Directionality	Enable as GPI Interrupt	
2	SDP2_IODIR	SDP2_GPIEN	27
1	SDP1_IODIR	SDP1_GPIEN	26
0	SDP0_IODIR	SDP0_GPIEN	25

- When the SDP pins are used as IEEE1588 auxiliary signals they can generate an interrupt on any transition (rising or falling edge), refer to [Section 7.7.4](#).
- A pair of SDPs can be used to create an I²C interface as described in [Section 3.4.1](#).

All SDP pins can be allocated to hardware functions. See more details on IEEE1588 auxiliary functionality in [Section 7.7.4](#) while I/O pins functionality are programmed by the TimeSync Auxiliary Control (TSAUXC) register.

If mapping of these SDP pins to a specific hardware function is not required, the pins can be used as general purpose software defined I/Os. For any of the function-specific usages, the SDP I/O pins should be set to native mode by software setting of the *SDPxxx_NATIVE* bits in the ESDP register. Native mode in those SDP I/O pins, defines the pin functionality at inactive state (reset or power down) while behavior at active state is controlled by the software. The hardware functionality of these SDP I/O pins differs mainly by the active behavior controlled by software.

Table 3-58 summarize the setup required to achieve each of the possible SDP per port configurations.

Table 3-58. SDP Settings

SDP	Usage	NVM Setting	ESDP			
			SDPx_NATIVE	SDPx_IODIR	SDP1_Function	SDP23_function
0	SDP	N/A	0	Input/Output	N/A	
	1588 functionality as defined by the TSSDP register	N/A	1	Input/Output		
1	SDP	N/A	0	Input/Output	0	N/A
	Reserved		N/A	N/A	N/A	
	1588 functionality as defined by the TSSDP register	N/A	1	Input/Output	0	
	Thermal Sensor	TS NVM-based Mode Enable bit in the Common Firmware Parameters word	0	Output	1	
	Reserved	N/A	1	N/A	1	
2	SDP	N/A	0	Input/Output	N/A	N/A
	1588 functionality as defined by the TSSDP register		1	Input/Output		0
	I ² C clock		1	N/A		1
3	SDP	N/A	0	Input/Output	N/A	N/A
	1588 functionality as defined by the TSSDP register		1	Input/Output		0
	I ² C data		1	N/A		1

3.4.1 I²C Over SDP

The I²C usage of SDP pins is enabled by setting the *SDP23_function* bit and the *SDP[23]_NATIVE* bits to 1b in ESDP register. This relates to the SDPx_2 and SDPx_3 pins, which operate as I2C_CLK and I2C_DATA, respectively.

The I²C interface operates via the I2CCMD and I2CPARAMS register set. Since this register set can be used by either software or firmware in alternation, its ownership must be acquired/released via the semaphore ownership taking/release flows described in [Section 4.7](#).

The I²C interface can be used in two methods, a hardware based access, where the device initiate a transaction following a software device driver request via the I2CCMD register or a software controlled bit-banging using the I2CPARAMS register.

3.4.1.1 Hardware Based I²C Access

The following flows should be used to access an I²C register.

As part of device initialization, or anytime before the actual access, the following parameters should be set:

- I2CPARAMS.PHYADD — the address of the device to access.
- I2CPARAMS.ACCESS_WIDTH — the width of the data to read or write (byte or word).

Note: The I2CPARAMS register should not be modified during an I²C transaction.

To execute a write access, the following steps should be done:

1. Check that register is ready: Poll I2CCMD.R bit until it is read as 1b.
2. Command — The I2CCMD register is initialized with the appropriate PHY register address in REGADD field, the data to write in the DATA field and the operation (write) to the OP field (0b).
 - a. If an interrupt is required, set the I2CCMD.I field
3. Check that command is done: Poll I2CCMD.R bit until it is read as 1b.
 - a. Check that no error is indicated in the I2CCMD.E field.

To execute a read access, the following steps should be done:

1. Check that register is ready: Poll I2CCMD.R bit until it is read as 1b.
2. Command — The I2CCMD register is initialized with the appropriate PHY register address in the REGADD field, and the operation (read) to the OP field (1b).
 - a. If an interrupt is required, set the I2CCMD.I field
3. Check that command is done: Poll I2CCMD.R bit until it is read as 1b.
 - a. Check that no error is indicated in the I2CCMD.E field.
4. Read the data returned from the I2CCMD.DATA field. If a byte access is done (I2CPARAMS.ACCESS_WIDTH = 0), only DATA[7:0] is valid.

See Section 3.4.1.3 below for the I²C commands supported when using the built in read and write commands. All the transactions uses a clock of 100 KHz. When using the bit-bang method any command can be given to the I²C device.

3.4.1.2 Bit-Bang Based I²C Access

In this mode, the software device driver controls the I²C interface directly using the I2CPARAMS register according to the following table:

Pad	Field Controlling the Output Value	Field Reflecting the Input Value	Field Controlling the Output Enable Value ¹
SDPx_2 (I ² C clock)	CLK_OUT	CLK_IN	CLK_OE_N
SDPx_3 (I ² C data)	DATA_OUT	DATA_IN	DATA_OE_N

1. 0b = Pad is output. 1b = Pad is input.

3.4.1.3 Supported Commands

Note: The gray columns below denotes cycles driven by the I²C device. White columns denotes cycles driven by the E610.

When a word read command (I2CPARAMS.ACCESS_WIDTH = 1b, I2CCMD.OP =1b) is given the following sequence is done by the E610:

Table 3-59. I²C Read Transaction - Dummy Write

1	7	1	1	8	1	
S	Device Address	Wr	A	Register Address	A	
	From I2CCMD.PHYADD	0	0	From I2CCMD.REGADD	0	

Table 3-60. I²C Read Transaction - Word Read

1	7	1	1	8	1	8	1	1
S	Device Address	Rd	A	Data	A	Data	A	P
	From I2CPARAMS.PHYADD	1	0	Stored in I2CCMD.DATA[7:0]	0	Stored in I2CCMD.DATA[15:8]	0	

When a byte read command (I2CPARAMS.ACCESS_WIDTH = 0b, I2CCMD.OP =1b) is given the following sequence is done by the E610:

Table 3-61. I²C Read Transaction - Dummy Write

1	7	1	1	8	1	
S	Device Address	Wr	A	Register Address	A	
	From I2CPARAMS.PHYADD	0	0	From I2CCMD.REGADD	0	

Table 3-62. I²C Read Transaction - Byte Read

1	7	1	1	8	1	1
S	Device Address	Rd	A	Data	A	P
	From I2CPARAMS.PHYADD	1	0	Stored in I2CCMD.DATA[7:0]	0	

When a word write command (I2CPARAMS.ACCESS_WIDTH = 1b, I2CCMD.OP =0b) is given the following sequence is done by the E610:

Table 3-63. I²C Write Transaction - Word Write

1	7	1	8	1		8	1	8	1	1
S	Device Address	Wr	Register Address	A		Data	A	Data	A	P
	From I2CPARAMS.PHYADD	0	From I2CCMD.REGADD	0		From in I2CCMD.DATA[7:0]	0	From in I2CCMD.DATA[15:8]	0	

When a byte write command (I2CPARAMS.ACCESS_WIDTH = 0b, I2CCMD.OP =0b) is given the following sequence is done by the E610:



Table 3-64. I²C Write Transaction - Byte Write

1	7	1	8	1	8	1	1
S	Device Address	Wr	Register Address	A	Data	A	P
	From I2CPARAMS.PHYADD	0	From I2CCMD.REGADD	0	From in I2CCMD.DATA[7:0]	0	

3.5 General Purpose I/O (GPIO) and LEDs

The E610 provides a total of 29 GPIOs used for SDP, 1588, LED, and other general purposes.

The E610 provides two types of GPIOs. Low latency GPIOs, that are directly controlled by the device registers and standard GPIOs that are provided by an I/O Widget. Four direct, low latency per port GPIOs are connected directly to the I/O pads and another 4 LEDs per port, as described below, have direct access to I/O pads as well to support legacy software.

In addition, the E610 controls an I/O Widget that provides interface to those GPIOs as well, and allows SDP, LED, and other general purpose interfaces.

Note: SDPs[23:16] are used for pin strapping. Other options for testing purposes are provide in [Table 3-66](#).

3.5.1 LEDs

3.5.1.1 Legacy LED Controlled by Software/Hardware

The E610 core implements four output drivers intended for driving external LED circuits per port. Each of the four LED outputs can be individually configured to select the particular event, state, or activity, which is indicated on that output. In addition, each LED can be individually configured for output polarity as well as for blinking versus non-blinking (steady-state) indication.

The configuration for LED outputs is specified via the LEDCTL register. Furthermore, the hardware-default configuration for all LED outputs can be specified via NVM fields ([Section 6.3.74.4](#) and [Section 6.3.74.5](#)) thereby supporting LED displays configurable to a particular OEM preference.

Each of the four LED's can be configured to use one of a variety of sources for output indication. For more information on the MODE bits see LEDCTL register.

The *IVRT* bits enable the LED source to be inverted before being output or observed by the blink-control logic. LED outputs are assumed to normally be connected to the negative side (cathode) of an external LED.

The *BLINK* bits control whether the LED should be blinked (on for 200 ms, then off for 200 ms or 83 ms on and 83 ms off according to the LEDCTL.GLOBAL_BLINK_MODE) while the LED source is asserted. The blink control can be especially useful for ensuring that certain events, such as ACTIVITY indication, cause LED transitions, which are sufficiently visible by a human eye.

Note: The LINK/ACTIVITY mode functions slightly differently than others as it ignores the BLINK value. The LED is:

- Off if there is no LINK
- On if there is LINK and no ACTIVITY
- Blinks if there is LINK and ACTIVITY

The mapping in Table 3-65 is used to specify the LED control source (MODE) for each LED output:

Table 3-65. LED Mapping

MODE	Selected Mode	Source Indication
0000b	LINK_UP	Asserted or blinking according to the <i>LEDx_BLINK</i> setting when any speed link is established and maintained.
0001b	LINK_10G	Asserted or blinking according to the <i>LEDx_BLINK</i> setting when a 10 Gb/s link is established and maintained.
0010b	MAC_ACTIVITY	Active when link is established and packets are being transmitted or received. In this mode, the <i>LEDx_BLINK</i> must be set.
0011b	FILTER_ACTIVITY	Active when link is established and packets are being transmitted or received that passed MAC filtering. In this mode, the <i>LEDx_BLINK</i> must be set.
0100b	LINK/ACTIVITY	Asserted steady when link is established and there is no transmit or receive activity. Blinking when there is link and receive or Transmit activity.
0101b	LINK_1G	Asserted or blinking according to the <i>LEDx_BLINK</i> setting when a 1 Gb/s link is established and maintained.
0110b	LINK_100	Asserted or blinking according to the <i>LEDx_BLINK</i> setting when a 100 Mb/s link is established and maintained.
0111b	LINK_2_5G	Asserted or blinking according to the <i>LEDx_BLINK</i> setting when a 2.5 Gb/s link is established and maintained.
1000b	LINK_5G	Asserted or blinking according to the <i>LEDx_BLINK</i> setting when a 5 Gb/s link is established and maintained.
1001b:1101b	Reserved	Reserved.
1110b	LED_ON	Always asserted or blinking according to the <i>LEDx_BLINK</i> setting.
1111b	LED_OFF	Always de-asserted.

3.5.1.2 I/O Widget SDP and LED

The E610 controls an I/O widget, which provides 8 SDPs and 12 LEDs. The I/O widget is accessed using register access commands.

The eight widget SDPs are used for Ethernet link management purposes, as follows:

- Directly controlling link topology components, such as external PHY or re-timer. This is mostly used in low port count applications (such as reset or interrupt line.)
- Detecting SFP module existence, in low port count applications (as MOD_ABS).

The 12 widget SDP/LED pins can be used as additional SDPs, for functions as described above or can be used as LEDs for low port count applications, up to three LEDs per port in 4-port configurations.

When a port has three dedicated LED pins they are configured as follows:

- One activity LED.
- One LED indicating the LINK is running at highest available speed (firmware only sets the highest Link speed in the LED control).
- One LED indicating the LINK is up but not at the highest speed available (firmware sets all speeds but the highest).

Another possible configuration is in a case of a break-out cable with QSFP+ cage. In this case there are only three LED pins for the entire cage that operates this way:

- One activity LED.
- One LED indicating that all LINKs in this cage are running at highest available speed (firmware configuration is TBD).
- One LED indicating the at least one LINK in this cage is up (firmware sets all speeds but the highest).

The link topology netlist define how to use the IO widget SDP and LEDs.

3.5.1.3 GPIO Pin Names

Table 3-66 summarize the GPIO pin names:

Table 3-66. E610 GPIO Pins

Pin Name	Func0	Func1	Control Register	Usage
SDP00	I/O Widget SDP/LED	LKP core P0 LED[0]	IOW/GPIO_CTL[0], LEDCTL, IO_CTRL_SDP_*SEL	LED or SDP assigned via the link topology netlist.
SDP01	I/O Widget SDP/LED	LKP core P0 LED[1]	OW/GPIO_CTL[1], LEDCTL, IO_CTRL_SDP_*SEL	
SDP02	I/O Widget SDP/LED	LKP core P0 LED[2]	W/GPIO_CTL[2], LEDCTL, IO_CTRL_SDP_*SEL	
SDP03	I/O Widget SDP/LED	LKP core P0 LED[3]	IOW/GPIO_CTL[3], LEDCTL, IO_CTRL_SDP_*SEL	
SDP04	I/O Widget SDP/LED	LKP core P1 LED[0]	IOW/GPIO_CTL[4], LEDCTL, IO_CTRL_SDP_*SEL	
SDP05	I/O Widget SDP/LED	LKP core P1 LED[1]	IOW/GPIO_CTL[5], LEDCTL, IO_CTRL_SDP_*SEL	
SDP06	I/O Widget SDP/LED	LKP core P1 LED[2]	IOW/GPIO_CTL[6], LEDCTL, IO_CTRL_SDP_*SEL	
SDP07	I/O Widget SDP/LED	LKP core P1 LED[3]	IOW/GPIO_CTL[7], LEDCTL, IO_CTRL_SDP_*SEL	
SDP08	I/O Widget	syncE_clock0	IOW/GPIO_CTL[8], IO_CTRL_SDP_*SEL	Recovered clocks (divided) for SyncE.
SDP09	I/O Widget	syncE_clock1	IOW/GPIO_CTL[9], IO_CTRL_SDP_*SEL	
SDP10	I/O Widget		IOW/GPIO_CTL[10], IO_CTRL_SDP_*SEL	
SDP11	I/O Widget		IOW/GPIO_CTL[11], IO_CTRL_SDP_*SEL	
SDP12	I/O Widget		IOW/GPIO_CTL[12], IO_CTRL_SDP_*SEL	
SDP13	I/O Widget		IOW/GPIO_CTL[13], IO_CTRL_SDP_*SEL	

Table 3-66. E610 GPIO Pins [continued]

Pin Name	Func0	Func1	Control Register	Usage
SDP14	I/O Widget		IOW/GPIO_CTL[14], IO_CTRL_SDP_*SEL	
SDP15	I/O Widget		IOW/GPIO_CTL[15], IO_CTRL_SDP_*SEL,	
SDP16	I/O Widget	LKP core P0 SDP[0]	IOW/GPIO_CTL[16], IO_CTRL_SDP_*SEL, ESDP	1588 timing support. can be used for 1588 / I ² C over SDP and software general purpose.
SDP17	I/O Widget	LKP core P0 SDP[1]	IOW/GPIO_CTL[17], IO_CTRL_SDP_*SEL, ESDP	
SDP18	I/O Widget	LKP core P0 SDP[2]	IOW/GPIO_CTL[18], IO_CTRL_SDP_*SEL, ESDP	
SDP19	I/O Widget	LKP core P0 SDP[3]	IOW/GPIO_CTL[19], IO_CTRL_SDP_*SEL, ESDP	
SDP20	I/O Widget	LKP core P1 SDP[0]	IOW/GPIO_CTL[20], IO_CTRL_SDP_*SEL, ESDP	1588 timing support.
SDP21	I/O Widget	LKP core P1 SDP[1]	IOW/GPIO_CTL[21], IO_CTRL_SDP_*SEL, ESDP	
SDP22	I/O Widget	LKP core P1 SDP[2]	IOW/GPIO_CTL[22], IO_CTRL_SDP_*SEL, ESDP	
SDP23	I/O Widget	LKP core P1 SDP[3]	IOW/GPIO_CTL[23], IO_CTRL_SDP_*SEL, ESDP	
SDP24	I/O Widget		IOW/GPIO_CTL[24], IO_CTRL_SDP_*SEL	PHY0_reset N.
SDP25	I/O Widget		IOW/GPIO_CTL[25], IO_CTRL_SDP_*SEL	PHY1_reset N.
SDP26	I/O Widget		IOW/GPIO_CTL[26], IO_CTRL_SDP_*SEL	PHY0_INT_N input.
SDP27	I/O Widget		IOW/GPIO_CTL[27], IO_CTRL_SDP_*SEL	PHY1_INT_N input.
SDP28	I/O Widget	UART_TXD	IOW/GPIO_CTL[28], IO_CTRL_SDP_*SEL	

3.6 Network Interface

3.6.1 Overview

The E610 provides dual-port network connectivity with copper media. Each port includes integrated MAC-PHY functionality and can be operated at either 10GBASE-T, 1GBASE-T, or 100BASE-T(X) link speed for copper or 10G, 1G and 100M bps for backplane media.

In terms of functionality there is no primary and secondary port as each port can be enabled or disabled independently from the other, and they can be set at different link speeds.

The integrated copper PHYs support the following specifications:

- 10GBASE-T as per the IEEE 802.3an standard.
- 1000BASE-T and 100BASE-TX as per the IEEE 802.3 standard.

Note: The reader is assumed to be familiar with the specifications included in these standards, which is not overlapping with content of subsequent sections.

All MAC configuration is performed using Device Control registers mapped into system memory or I/O space; an internal MDIO/MDC interface is used to configure the PHY operation.

3.6.2 Internal MDIO Interface

The E610 implements IEEE 802.3 Management Data Input/Output Interface (MDIO Interface or MII Management Interface) between each MAC and its attached integrated PHY. This interface provides firmware and software the ability to monitor and control the state of the PHY. It provides indirect access to an internal set of addressable PHY registers. It complies with the new protocol defined by Clause 45 of IEEE 802.3 std. No backward compliance with Clause 22.

Notes: MDIO access to PHY registers must be operational from the time the PHY has completed its initialization once having read the PHY image from the NVM.

During internal PHY reset events where the MAC is not reset, PHY registers might not be accessible and the MDIO access does not complete.

Software is notified that PHY initialization and/or reset has completed by either polling or by PHY reset done interrupt.

The internal MDIO interface is accessed through registers MSCA and MSRWD. An access transaction to a single PHY register is performed by setting the MSCA.MDICMD bit to 1b after programming the appropriate fields in the MSCA and MSRWD registers. The MSCA.MDICMD bit is auto-cleared after the read or write transaction completes.

To execute a write access, the following steps should be done:

1. Address Cycle - Register MSCA is initialized with the appropriate PHY register address in *MDIADD DEVADD*, and *PORTADD* fields, the *OPCODE* field set to 00b and *MDICMD* bit set to 1b.
2. Poll MSCA.MDICMD bit until it is read as 0b.
3. Write Data Cycle - Data to be written is programmed in field MSRWD.MDIWRDATA.

4. Write Command Cycle - *OPCODE* field in the MSCA register is set to 01b for a write operation and the MSCA.*MDICMD* bit is set to 1b.
5. Wait for the MSCA.*MDICMD* bit to reset to 0b, which indicates that the transaction on the internal MDIO interface completed.

To execute a read access, the following steps should be done:

1. Address Cycle - Register MSCA is initialized with the appropriate PHY register address in *MDIADD DEVADD*, and *PORTADD* fields, the *OPCODE* field set to 00b and *MDICMD* bit set to 1b.
2. Poll MSCA.*MDICMD* bit until it is read as 0b.
3. Read Command Cycle - *OPCODE* field in the MSCA register is set to 11b for a read operation and the MSCA.*MDICMD* bit is set to 1b.
4. Wait for the MSCA.*MDICMD* bit to reset to 0b, which indicates that the transaction on the internal MDIO interface completed.
5. Read Data Cycle - Read the data in field MSRWD.*MDIRDDATA*.

Notes: A read-increment-address flow is performed if the *OPCODE* field is set to 10b in [Step 3](#). The address is increased internally once data is read at [Step 5](#) so that no address cycle is needed to perform a data read from the next address.

Before writing the MSCA register, make sure that the MDIO interface is ready to perform the transaction by reading MSCA.*MDICMD* as 0b.

3.6.3 Enabling the Network Link

In most use cases it is desired that The E610 is powered up with active link. This option is needed for APM wake on LAN (described in [Section 5.4.1](#)) or Manageability connection to the network. Such an option is requested also by customers that expect to see active link LEDs when powering the system even pre OS. On the other hand, there are customers who do not care about all of the above, expecting the device to be initialized with no link. Possible use cases are for power saving or due to strict security reasons.

The E610 addresses both options. As part of the device initialization flow, The E610 checks the value of the EMP_PORT_DIS flag in the PRTGEN_CNF register (loaded from the NVM). If it is cleared (enable), then the firmware enables the Ethernet link. If the EMP_PORT_DIS flag is active, then the link is not established until the software driver enables it.

As part of the software driver initialization flow, it is expected to check the PORT_DIS and the EMP_PORT_DIS flags. If the port is enabled by the PORT_DIS flag but disabled by the EMP_PORT_DIS flag, the software can initiate the "Setup Link and Restart Auto-Negotiation" admin command to enable the link. As a response, the EMP clears the EMP_PORT_DIS flag and enables the Ethernet link.

3.6.4 Integrated Internal PHY (PHYSS) Functionality

3.6.4.1 Introduction

The E610 provides two Ethernet port interfaces. The following sections describe the physical interfaces of the internal SERDES that The E610 support and manage through firmware.

Table 3-67. Supported Electrical Modes

Speed	Mode	Spec	Comments
10G	10GBASE-KR	PMD: • IEEE 802.3 Clause 72 PCS: • IEEE 802.3 Clause 49	• Used for Backplane connectivity.
	SFI	PMD: • SFF 8431 PCS: • IEEE 802.3 Clause 49	• Used for connecting to SFP+ module. • limiting mode for optical & active cables. • Appendix E linear mode for passive copper twin-ax.
	USXGMII	USXGMII spec	• Used for chip-to-chip connectivity single port.
5G	USXGMII (10G)	USXGMII spec	• Used for chip-to-chip connectivity single port.
2.5G	USXGMII (10G)	USXGMII spec	• Used for chip-to-chip connectivity single port.
1G	1000BASE-KX	PMD: • IEEE 802.3 Clause 70 PCS: • IEEE 802.3 Clause 36	• Used for Backplane connectivity.
	SGMII	PMD: • Cisco SGMII spec PCS: • IEEE 802.3 Clause 36, 37 like	• Used for chip-to-chip connectivity.
	USXGMII (10G)	USXGMII spec	• Used for chip-to-chip connectivity single port.
100M	SGMII	PMD: • SGMII spec PCS: • IEEE 802.3 Clause 36, 37 like	• Used for chip-to-chip connectivity.

3.6.4.2 Supported Speed Modes

The following table describe the different PHY modes of operation, speeds

Table 3-68. Supported Speed Modes

Media	Speed Mode	AN	FEC	PCS Core I/F	XGMII Average Rate (MHz)	XGMII Max Rate (MHz)	SERDES Rate (GHz)	PMA Width	PMA Rate (MHz)	PCS Module	Comments
KR	10G	73	74	XGMII	312.5	322.265625	10.3125	32	322.265625	XPCS	Confirm PMA width.
SFI-DA	10G	No	N/A	XGMII	312.5	322.265625	10.3125	32	322.265625	XPCS	
USXGMII	10G	37	N/A	XGMII	312.5	322.265	10.3125	32	322.265625	XPCS	
USXGMII	5G	37	N/A	XGMII	156.25	161.132	10.3125	32	322.265625	XPCS	
USXGMII	2.5G	37	N/A	XGMII	78.125	80.566	10.3125	32	322.265625	XPCS	
USXGMII	1G	37	N/A	XGMII	31.25	32.226	10.3125	32	322.265625	XPCS	
USXGMII	100M	37	N/A	XGMII	3.125	3.226	10.3125	32	322.265625	XPCS	
KX	1G	73	N/A	XGMII	31.25	31.25	1.25	10	125	XPCS	
BASE-X	1G	No	N/A	XGMII	31.25	31.25	1.25	10	125	XPCS	No LT.
SGMII	2.5G	No	N/A	XGMII	78.125	78.125	3.125	10	312.5	XPCS	
SGMII	1G	37	N/A	XGMII	31.25	31.25	1.25	10	125	XPCS	Nice to have.
SGMII	100M	37	N/A	XGMII	3.125	3.125	1.25	10	125	XPCS	

3.6.4.3 MAC - PHY Interface

The E610 provides up to two Ethernet ports running on XGMII 32 bit interface for all supported speed and media. (See IEEE 802.3 clause 46)

3.6.4.4 Firmware PHY Management

All communication between the LAN controller's firmware engine to the PHY is done over SB-IOSF read and write messages (see SB-IOSF section). Using the SB-IOSF, the firmware can read and write information to the PHY block CSRs.

This bidirectional communication channel, using SB-IOSF read and write commands, is used to manage the following scenarios:

1. Firmware uses the SB-IOSF to configure the PHY and control the link parameters.
2. Following a new link up event, firmware read the PHY's negotiated attributes as well as the link partner advertised information in order to learn the status of:
 - a. Pause capabilities
 - b. Speed capabilities
 - c. EEE capabilities
 - d. FEC support
3. When the LAN controller is placed into low power state firmware may trigger LPLU flow.
4. When link should be powered down firmware can communicate the required power state to the PHY.

5. Following an internal reset event, firmware will signal a restart link procedure.

3.6.4.4.1 PHYSS Power-Up Support

During power on, the LAN's firmware plays an important role in bringing up the PHY core.

The following flow describes the power up sequence in general terms, the full timing details of this sequence are provided in the Initialization section.

PHYSS configuration is done in two stages:

- PHYSS SERDES SRAM firmware load.
- PHYSS PCS and link configurations.

The following describe the high level flow of PHYSS configuration:

1. PHYSS IOSF SB reset is de asserted
2. SERDES firmware (SRAM) down load should be done over IOSF-SB once firmware complete loaded and authenticated.
3. PHY reset.
4. When internal topology and LM are ready and software link up request was issued (AQC HICR), Configure the link parameters and ANEG attributes (FC, FEC, EEE, Speed).
5. PHYSS will start the link bring up operation.
6. LM/PM receives an INT from PHYSS.
7. LM/PM reads the interrupt source PHYSS register PHYSS_ICR_0/1_REG and ANEG results (FC, FEC, EEE + LP advertised information).
8. Link will come up.
9. Firmware will reports the link state to the software driver.

3.6.4.4.2 PHY Link Events

During link management (LM), the link management firmware might trigger events for the internal PHY, as described in the following table.

Table 3-69. PHY Link Events

Event	Description	Impact on PHYSS
Link reset	Performed during regular firmware flow when ever link state needs to be reset	• This will reset the PHYSS state but will not affect the PCS configuration. • Link will drop and the link state machine will be restarted.
Link disable	Performed when link is not needed due to configuration or low power state	• This will reset the PHYSS's state but will not affect the PCS configuration. • Link will drop and the link state machine will be paused. • This is not a power down control
Power down	Performed when link is not needed due to configuration or low power state	• This will reset the PHYSS state but will not affect the PCS configuration. • Link will drop and the link state machine will be paused. • PHYSS may take further action to power down its link

3.6.4.4.3 PHYSS SRAM Load

Basic assumption is that PHY SRAM load is required only after PHY reset. the SRAM content should not change during time for each port / lane.

The default state of the PHY is to run through SRAM_bootload_byp flow which the SERDES hardware default are not loaded into the SRAM, but only the firmware side will load entire SRAM image. this is done in a dedicated access to the SRAM address space and not through the PCS. so, SRAM load is not dependent on PCS and PHY state.

Steps:

1. Turn off SRAM clock - Set PHYSS_SERDES_SRAM_MUX_SEL.gate_clk to "1" to guarantee glitch free switching.
2. Set PHYSS_SERDES_SRAM_MUX_SEL bit [0] to "1" to switch to SB domain.
3. Turn on SRAM clock - Set PHYSS_SERDES_SRAM_MUX_SEL.gate_clk to "0" t.
4. Download SRAM image data- write over IOSF SB RDA in DW resolution.
5. Turn off SRAM clock.
6. Set PHYSS_SERDES_SRAM_MUX_SEL bit [0] to "0" to switch to PHY domain.
7. Turn on SRAM clock.

Note: The above steps should repeated each time SRMA load is required during run time.

Once SRAM image done, firmware will de assert the PHY reset by write "1" to the PHYSS_LANE_RST_N register with the corresponding value in the PHYSS_SUBSYSTEM_LANE_SEL.

SRAM_INIT_DONE indication should be set immediately on PCS.

To complete process and start SERDES running from SRAM, firmware should set the EXT_LD_DONE bit inside the PCS.

3.6.4.4.4 PHYSS SyncE Support

PHYSS provides recovered clock per lane for SyncE purpose. the clock will run the PMA interface higher rate. For that a SRCP block will divided those clocks to desired frequency and provide it through GPIOs SDP08 and SDP09 to an external DPLL on the board. then, after one of these clocks was filtered and cleaned by external DPLL, it will provide it back as a reference clock to SERDES through Eth_clk_P/N.

3.6.4.4.5 Time Stamp Indication

In order to get more accurate time stamp close to network line, a start of packet indication for each TX and RX packets per port can be provide from PHYSS internal stages to the time stamp point as a configuration option.

3.6.4.4.6 PHYSS Power-Up Clocking

As describe in clocking chapter 4.3, the source for SERDES reference clock 156.25 MHz can be external by the ETH_CLK_P/N pins or by local top PLL which connected to the ref_alt0_clk_p/m CML pins of the SERDES.

Note: The ref_alt0_clk are running on VP (0.9 V) level.

3.6.5 Ethernet Flow Control (FC)

The E610 supports flow control as defined in 802.3x, as well as the specific operation of asymmetrical flow control defined by 802.3z. The E610 also supports Priority Flow Control (PFC), known as Class Based Flow Control, as part of the DCB architecture.

Note: The E610 can either be configured to receive regular FC packets or PFC packets. The E610 does not support receiving both types of packets simultaneously.

FC is implemented to reduce receive buffer overflows, which result in the dropping of received packets. FC also allows for local controlling of network congestion levels. This can be accomplished by sending an indication to a transmitting station of a nearly full receive buffer condition at a receiving station.

The implementation of asymmetric FC allows for one link partner to send FC packets while being allowed to ignore their reception. For example, not required to respond to PAUSE frames.

The following registers are defined for implementing FC. In DCB mode, some of the registers are duplicated replicated per Traffic Class (TC), up to eight duplicates copies of the registers. If DCB is disabled, index [0] of each register is used.

- MAC Flow Control Register (*MFLCN*) — Enables FC and passing of control packets to the host.
- Flow Control Configuration (*FCCFG*) — Determines mode for Tx FC (No FC vs. link-based vs. priority-based). Note that if Tx FC is enabled, Tx CRC by hardware should be enabled as well (*HLREG0.TXCRCEN* = 1b).
- Flow Control Source Address Low, High (*RAL*[0], *RAH*[0])
- Flow Control Destination Address Low, High (*FCAMACL*, *FCAMACH*) — 6-byte FC multicast address.
- Priority Flow Control Type Opcode (*PFCTOP*) — Contains the type and opcode values for PFC.
- Flow Control Receive Threshold High (*FCRTH*[7:0]) — A set of 13-bit high watermarks indicating receive buffer fullness. A single watermark is used in link FC mode and up to eight watermarks are used in PFC mode.
- Flow Control Receive Threshold Low (*FCRTL*[7:0]) — A set of 13-bit low watermarks indicating receive buffer emptiness. A single watermark is used in link FC mode and up to eight watermarks are used in PFC mode.
- Flow Control Transmit Timer Value (*FCTTV*[3:0]) — A set of 16-bit timer values to include in transmitted PAUSE frame. A single timer is used in link FC mode and up to eight timers are used in PFC mode.
- Flow Control Refresh Threshold Value (*FCRTV*) — 16-bit PAUSE refresh threshold value (in legacy FC *FCRTV*[0] must be smaller than *FCTTV*[0]).

3.6.5.1 MAC Control Frames and Reception of Flow Control Packets

3.6.5.1.1 MAC Control Frame — Other than FC

The IEEE specification reserved the EtherType value of 0x8808 for MAC control frames, which are listed in [Table 3-70](#).

Table 3-70. MAC Control Frame Format

Field	Description
DA	The <i>Destination Address</i> field can be an individual or multicast (including broadcast) address. Permitted values for the <i>Destination Address</i> field can be specified separately for a specific control opcode such as FC packets.
SA	Port Ethernet MAC Address (six bytes).
Type	0x8808 (two bytes).
Opcode	The MAC control opcode indicates the MAC control function.
Parameters	The MAC control <i>Parameters</i> field must contain MAC control Opcode-specific parameters. This field can contain none, one, or more parameters up to a maximum of minFrameSize = 20 bytes.
Reserved field = 0x00	The <i>Reserved</i> field is used when the MAC control parameters do not fill the fixed length MAC control frame.
CRC	Four bytes.

3.6.5.1.2 Structure of 802.3X FC Packets

802.3X FC packets are defined by the following three fields (see [Table 3-71](#)):

1. A match on the six-byte multicast address for MAC control frames or a match to the station address of the device (Receive Address Register 0). The 802.3x standard defines the MAC control frame multicast address as 01-80-C2-00-00-01.
2. A match on the *Type* field. The *Type* field in the FC packet is compared against an IEEE reserved value of 0x8808.
3. A match of the MAC control *Opcode* field has a value of 0x0001.

Frame-based FC differentiates XOFF from XON based on the value of the PAUSE *Timer* field. Non-zero values constitute XOFF frames while a value of zero constitutes an XON frame. Values in the *Timer* field are in units of pause quanta (such as slot time). A pause quanta lasts 64 byte times, which is converted into an absolute time duration according to the line speed.

Note: XON frame signals the cancellation of the pause from that was initiated by an XOFF frame. Pause for zero pause quanta.

Table 3-71. 802.3X Packet Format

Field	Description
DA	01_80_C2_00_00_01 (6 bytes).
SA	Port Ethernet MAC Address (6 bytes).
Type	0x8808 (2 bytes).
Opcode	0x0001 (2 bytes).
Time	XXXX (2 bytes).
Pad	42 bytes.
CRC	4 bytes.

3.6.5.1.3 Priority Flow Control

DCB introduces support for multiple TCs assigning different priorities and bandwidth per TC. Link-level Flow Control (PAUSE) stops all the TCs. Priority Flow Control (PFC), known as Class Based Flow Control or CBFC, allows more granular Flow Control on the Ethernet link in a DCB environment as opposed to the PAUSE mechanism defined in 802.3X.

PFC is implemented to prevent the possibility of receive packet buffers overflow. Receive packet buffers overflow results in the dropping of received packets for a specific TC. Implement PFC by sending a timer indication to the transmitting station TC (XOFF) of a nearly full receive buffer condition at the E610. At this point the transmitter stops transmitting packets for that TC until the XOFF timer expires or a XON message is received for the stopped TC.

Similarly, once the E610 receives a priority-based XOFF it stops transmitting packets for that specific TC until the XOFF timer expires or XON packet for that TC is received.

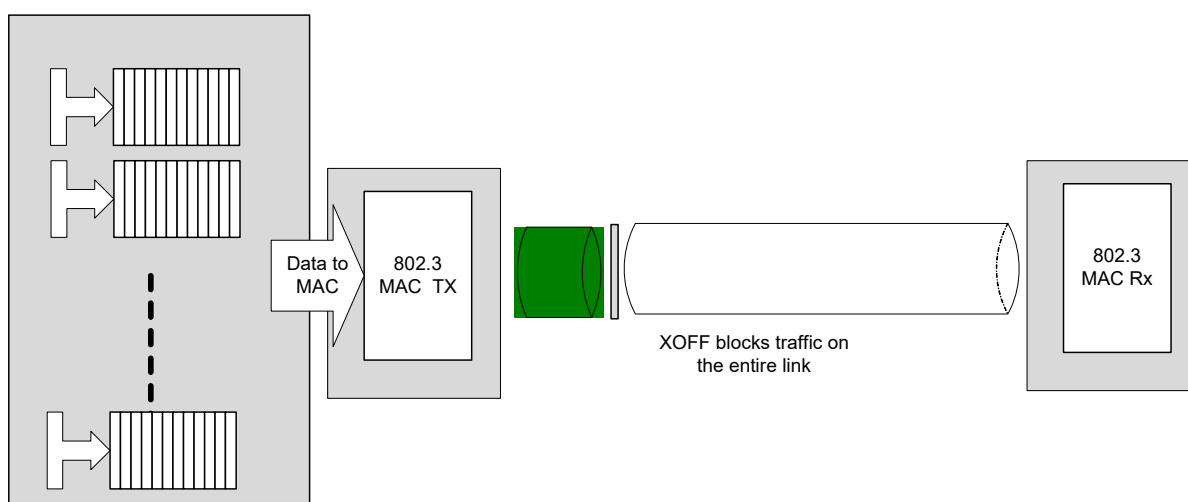


Figure 3-16. 802.3X Link Flow Control (PAUSE)

Link flow control (802.3X) causes all traffic to be stopped on the link. DCB uses the same mechanism of FC but provides the ability to do PFC on TCs, as shown in [Figure 3-17](#).

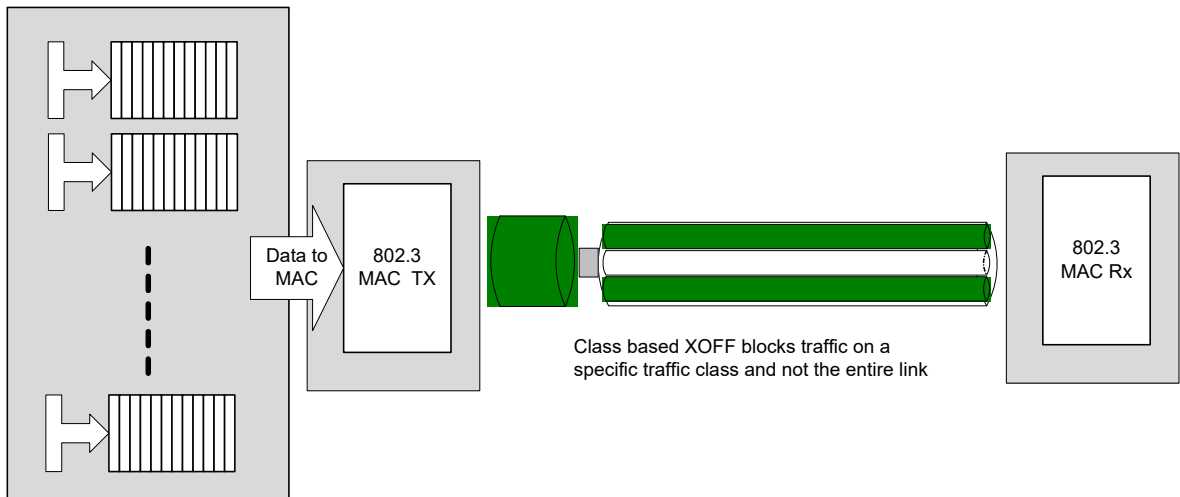


Figure 3-17. Priority Flow Control

Table 3-72. Packet Format for PFC

Field	Description
DA	01_80_C2_00_00_01 (6 bytes).
SA	Port Ethernet MAC Address (6 bytes).
Type	0x8808 (2 bytes).
Opcode	0x0101 (2 bytes).
Priority Enable Vector	0x00XX (2 bytes).
Timer 0	XXXX (2 bytes).
Timer 1	XXXX (2 bytes).
Timer 2	XXXX (2 bytes).
Timer 3	XXXX (2 bytes).
Timer 4	XXXX (2 bytes).
Timer 5	XXXX (2 bytes).
Timer 6	XXXX (2 bytes).
Timer 7	XXXX (2 bytes).
Pad	26 bytes.
CRC	4 bytes.

Table 3-73. Format of Priority Enable Vector

	ms octet	ls octet
Priority enable vector definition	0	e[7]...e[n]...e[0]
e[n] =1 => time (n) valid e[n] =0 => time (n) invalid		

The Priority Flow Control Type Opcode (PFCTOP) register contains the type and opcode values for PFC. These values are compared against the respective fields in the received packet.

Each of the eight timers refers to a specific User Priority (UP), such as Timer 0 refers to UP 0, etc. The E610 binds a UP and timer to one of its TCs according to the UP-to-TC binding tables. Refer to the RTTUP2TC register for the binding of received PFC frames to Tx TCs, and to the RTRUP2TC register for the binding of transmitted PFC frames to Rx TCs.

Tx manageability traffic is bound to one the TCs via the MNGTXMAP register, and should thus be paused according to RTTUP2TC mapping when receiving PFC frames.

When a PFC frame is formatted by the E610, the same values are replicated into every *Timer* field and priority enable vector bit of all the UPs bound to the associated TC. These values are configured in the RTRUP2TC register.

The following rule is applicable for the case of multiple UPs that share the same TC (as configured in the RTTUP2TC register). When PFC frames are received with different timer values for the previously mentioned UPs, the traffic on the associated TC must be paused by the highest XOFF timer's value.

3.6.5.1.4 Operation and Rules

The E610 operates in either link FC or in PFC mode. Note that enabling both modes concurrently is not allowed:

- Link FC is enabled by the *RFCE* bit in the MFLCN register.
- PFC is enabled per UP by the corresponding *RPFCE* bit in the MFLCN register, and globally by MFLCN.*RPFCM* bit.

Note: Link FC capability must be negotiated between link partners via the auto-negotiation process. The PFC capability is negotiated via some higher level protocol and the resolution is usually provided to the driver by the DCB management agent. It is the driver's responsibility to reconfigure the link FC settings (including *RFCE* and *PRFCE*) after the auto-negotiation process was resolved.

Note: Receiving a link FC frame while in PFC mode might be ignored or might pause TCs in an unpredictable manner. Receiving a PFC frame while in link FC mode is ignored. Flow control events that are ignored do not increment any flow control statistics counters.

Once the receiver has validated the reception of an XOFF, or PAUSE frame, the device performs the following:

- Increment the appropriate statistics register(s)
- Initialize the pause timer based on the packet's PAUSE *Timer* field (overwriting any current timer's value)
 - For PFC, this is done per TC. If several UPs are associated with a TC, the device sets the timer to the maximum value among all enabled *Timer* fields associated with the TC.
- Disable packet transmission or schedule the disabling of transmission after the current packet completes.
 - For PFC, this is done per paused TC
 - Tx manageability traffic is bound to a specific TC as defined in the MNGTXMAP register, and is thus paused when its TC is paused

Resumption of transmission can occur under the following conditions:

- Expiration of the PAUSE timer
 - For PFC, this is done per TC

- Receiving an XON frame (a frame with its PAUSE timer set to 0b)
 - For PFC, this is done per TC

Both conditions clear the relevant TXOFF status bits in the Transmit Flow Control Status (TFCS) register and transmission can resume. Hardware records the number of received XON frames.

3.6.5.1.5 Timing Considerations

When operating at 10 GbE line speed, the E610 must not begin to transmit a (new) frame more than 74 pause quanta after receiving a valid Link XOFF frame, as measured at the wires (a pause quantum is 512 bit times).

When operating at 1 GbE line speed, the E610 must not begin to transmit a (new) frame more than 2 pause quanta after receiving a valid Link XOFF frame, as measured at the wires.

When operating at 100 Mb/s line speed, the E610 must not begin to transmit a (new) frame more than 1 pause quantum plus 64 bit times after receiving a valid Link XOFF frame, as measured at the wires.

The 802.1Qbb draft 2, proposes that the tolerated response time for Priority XOFF frames are the same as Link XOFF frames with extra budget of 2 pause quanta. This extra budget is aimed to compensate the fact that decision to stop new transmissions from a specific TC must be taken earlier in the transmit data path than for the Link Flow Control case.

3.6.5.2 PAUSE and MAC Control Frames Forwarding

Two bits in the Receive Control register control transfer of PAUSE and MAC control frames to the host. These bits are *Discard PAUSE Frames (DPF)* and *Pass MAC Control Frames (PMCF)*. Note also that any packet must pass the L2 filters as well.

- The *DPF* bit controls the transfer of PAUSE packets to the host. The same policy applies to both link FC and PFC packets as listed in [Table 3-74](#). Note that any packet must pass the L2 filters as well.
- The *Pass MAC Control Frames (PMCF)* bit controls the transfer of non-PAUSE packets to the host. Note that when link FC frames are not enabled (RFCE = 0b), link FC frames are considered as MAC control frames for this case. Similarly, when PFC frames are not enabled (RPFCM = 0b), PFC frames are considered as MAC control frames as well.

Note: When virtualization is enabled, forwarded control packets are queued according to the regular switching procedure defined in [Section 7.8.3.4](#).

Table 3-74. Transfer of PAUSE Packet to Host (DPF Bit)

RFCE	RPFCM	DPF	Link FC Handling	PFC Handling
0b	0b	X	Treat as MAC control (according to <i>PMCF</i> setting).	Treat as MAC control (according to <i>PMCF</i> setting).
1b	0b	0b	Accept.	Treat as MAC control (according to <i>PMCF</i> setting).
1b	0b	1b	Reject.	Treat as MAC control (according to <i>PMCF</i> setting).
0b	1b	0b	Treat as MAC control (according to <i>PMCF</i> setting).	Accept.
0b	1b	1b	Treat as MAC control (according to <i>PMCF</i> setting).	Reject.
1b	1b	X	Unsupported setting.	Unsupported setting.

3.6.5.3 Transmitting PAUSE Frames

The E610 generates PAUSE packets to ensure there is enough space in its receive packet buffers to avoid packet drop. The E610 monitors the fullness of its receive FIFOs and compares it with the contents of a programmable threshold. When the threshold is reached, the E610 sends a PAUSE frame. The E610 supports both link FC and PFC — but not both concurrently. When DCB is enabled, it only sends PFC, and when DCB is disabled, it only sends link FC.

Note: Similar to receiving flow control packets previously mentioned, software can enable FC transmission by setting the FCCFG.TFCE field only after it is negotiated between the link partners (possibly by auto-negotiation).

3.6.5.3.1 Priority Flow Control (PFC)

The E610 operates in either a link 802.3X compliant mode or in a Priority Flow Control mode, but not in both at the same time.

The same watermarks mechanism is used for PFC and for 802.3X FC to determine when to send XOFF and XON packets. When PFC is used in the receive path, priority PAUSE packets are sent instead of 802.3X PAUSE packets. The format of priority PAUSE packets is described in [Section 3.6.5.1.3](#).

Specific considerations for generating PFC packets:

- When a PFC packet is sent, the packet sets all the UPs that are associated with the relevant TC (UP-to-TC association in receive is defined in RTRUP2TC register).

3.6.5.3.2 Operation and Rules

The TFCE field in the Flow Control Configuration (FCCFG) register enables transmission of PAUSE packets as well as selects between the link FC mode and the PFC mode.

The content of the Flow Control Receive Threshold High (FCRTH) register determines at what point the E610 transmits the first PAUSE frame. The E610 monitors the fullness of the receive FIFO and compares it with the contents of FCRTH. When the threshold is reached, the E610 sends a PAUSE frame with its pause time field equal to FCTTV.

At this time, the E610 starts counting an internal shadow counter (reflecting the pause time-out counter at the partner end). When the counter reaches the value indicated in FCRTV register, then, if the PAUSE condition is still valid (meaning that the buffer fullness is still above the low watermark), an XOFF message is sent again.

Once the receive buffer fullness reaches the low water mark, the E610 sends an XON message (a PAUSE frame with a timer value of zero). Software enables this capability with the XONE field of the FCRTL.

The E610 sends a PAUSE frame if it has previously sent one and the FIFO overflows. This is intended to minimize the amount of packets dropped if the first PAUSE frame did not reach its target.

3.6.5.3.3 Flow Control High Threshold — FCRTH

The E610 sends a PAUSE frame when the Rx packet buffer is full above the high threshold. The threshold should be large enough to overcome the worst case latency from the time that crossing the threshold is sensed until packets are not received from the link partner.

Referring to Annex O of IEEE802.1Qbb rev 2.3, worst case latency depends on **two** parameters:

1. Maximum frame size over the traffic class for which FCRTH is computed. It is referred as MaxFrame(TC).
2. Maximum frame size over the link (all traffic classes altogether). It is referred as MaxFrame(link).

Three values are envisaged for MaxFrame:

- 1.5 KB (Ethernet - Jumbo disabled)
- 9.5 KB (Jumbo enabled)

Worst case latency, which is referred as Standard Delay Value (Std DV), is given by:

$$\text{Std DV} = \text{MaxFrame(TC)} + \text{MaxFrame(link)} + \text{PFC Frame} + 2 \times \text{Cable Delay} + 2 \times \text{Interface Delay} + \text{Higher Layer Delay} \times \text{Sec Y Transmit Delay}$$

$$\text{Std DV (bit time units)} = \text{MaxFrame(TC)} + \text{MaxFrame(link)} + 672 + 2 \times 5,556 + 2 \times (25,600 + 8,192 + 2 \times 2,048) + 6,144 \times (\text{MaxFrame(link)} + 3,200)$$

MaxFrame(TC) term and MaxFrame(link) term included in Sec Y Transmit Delay correspond to worst case scenarios issued by the link partner. All other terms in Std DV formula must take in account worst case incoming traffic pattern which would lead to worst case buffer utilization as per the internal architecture of Rx packet buffer in the E610.

Internal architecture of the Rx packet buffer has the following restrictions:

1. Any packet starts at 32 byte aligned address.
2. Any packet has an internal status of 32 bytes. As a result, the Rx packet buffer is used at worst conditions when the Rx packet includes 65 bytes that are posted to the host memory. Assuming that the CRC bytes are not posted to host memory, in the worst case the Rx packet buffer can be filled at 1.44 higher rate than the wire speed (69-byte packet including CRC + 8-byte preamble + 12-byte back-to-back IFS consumes 4 x 32 bytes = 128 bytes on the Rx packet buffer).
3. An additional packet from the concerned traffic class may be inserted into the Rx packet buffer due to the internal loopback switch *just before* it is decided to issue XOFF to the link partner

It leads to the below revised formula for the E610:

$$\text{E610 DV (bit time units)} = 1.44 \times [(\text{MaxFrame(link)} + 672 + 2 \times 5,556 + 2 \times (25,600 + 8,192 + 2 \times 2,048) + 6,144 \times (3,200)] + \text{MaxFrame(TC)} + \text{MaxFrame(TC)} \times \text{MaxFrame(link)}$$

FCRTH must be set to the size of the Rx packet buffer allocated to the traffic class minus the E610 DV.

Table 3-75. E610 Delay Values (DV) Used for FCRTH

9.5 KB Jumbo Enabled	FCoE Traffic Class	E610 DV
No	No	24 KB
No	Yes	25 KB
Yes	No	50 KB
Yes	Yes	35 KB
No	No	27 KB

Table 3-75. E610 Delay Values (DV) Used for FCRTTH [continued]

9.5 KB Jumbo Enabled	FCoE Traffic Class	E610 DV
No	Yes	27 KB
Yes	No	60 KB
Yes	Yes	45 KB

3.6.5.3.4 FC Low Threshold — FCRTL

The low threshold value is aimed to protect against wasted available host bandwidth. There is some latency from the time that the low threshold is crossed until the XON frame is sent and packets are received from the link partner. The low threshold must be set high enough so that the Rx packet buffer does not get empty before any new entire packets are received from the link partner. When considering data movement from the Rx packet buffer to host memory, large packets represent the worst. Assuming the host bandwidth is about the bandwidth on the wire (when dual ports are active at a given time), and assuming a PCIe round trip is required to get the receive descriptors, we get the following formula for FCRTL:

$$\text{FCRTL} = 2 \times \text{MaxFrame}(\text{TC}) + \text{PCIe round trip delay}$$

PCIe round trip delay is assumed to be $\sim 1 \mu\text{s}$ and it must cover for worst case incoming traffic pattern (buffer utilization by 1.44 than wire rate):

$$\text{FCRTL (bit time units)} = 2 \times \text{MaxFrame}(\text{TC}) + 1.44 \times 10,000$$

Setting the FCRTL to lower values than expressed by the previous equation is permitted. It might simply result with potential sub-optimal use of the PCIe bus once bandwidth is available.

Table 3-76. E610 FCRTL

9.5 KB Jumbo Enabled	FCoE Traffic Class	E610 DV
No	No	5 KB
No	Yes	7 KB
Yes	No	21 KB
Yes	Yes	7 KB

3.6.5.3.5 Packet Buffer Size

When FC is enabled, the total size of a TC packet buffer must be large enough for the Low and high thresholds. To avoid constant transmission of XOFF and XON frames it is recommended to add some space for hysteresis type of behavior. The difference between the two thresholds is recommended to be at least one frame size (when 9.5 KB jumbo frames are expected over the TC) and larger than a few frames in other cases (4.5 KB for instance). If the available Rx buffer is large enough, it is recommended to increase as much as possible the hysteresis budget. If the available Rx buffer is not large enough it might be required to cut both the low threshold as well as the hysteresis budget.

- For a PFC-enabled TC:
 - FCRTH = FCRTL + hysteresis budget = FCRTL + Max (MaxFrame(TC), $4.5 \times 1024 \text{ B}$)
 - Rx Packet Buffer size = FCRTH + the E610 DV (see [Section 3.6.5.3.3](#))
- For a best effort TC:
 - Rx Packet Buffer size = FCRTL, as the same considerations than described in [Section 3.6.5.3.4](#) play here to avoid bubbles over PCIe

The total Rx Packet Buffer size available to a port for all its supported TCs is either 384 KB, 320 KB, or 256 KB, depending on the size allocated to the Flow Director table, 0 KB, 64 KB, or 128 KB, respectively.

Table 3-77 assumes four PFC-enabled Traffic Classes are defined over the port, of which two are allocated to traffic and two for other loss less traffic types like iSCSI, etc. The table lists the recommended settings for the supported combinations. When less than 4 PFC-enabled TCs are defined, and/or when less than 8 TCs are defined, it is recommended to refer to the setting rules described in this section, in [Section 3.6.5.3.3](#), and in [Section 3.6.5.3.4](#). Note that reducing the number of TCs of a port to what is really needed, helps increasing the port's throughput.

Table 3-77. Some Recommended Rx Packet Buffer Settings

Flow Director Table Size	9.5 KB Jumbo Enabled	Packet Buffer Size of Any of the 4 Best Effort TCs	Packet Buffer Size of Any of the 2 FCoE TCs	Packet Buffer Size of Any of the Other 2 PFC-Enabled TCs
No	Yes	33 KB	62 KB FCRTL = 7 KB FCRTH = 37 KB	61 KB FCRTL = 6 KB FCRTH = 37 KB
No	Yes	27 KB	52 KB FCRTL = 7 KB FCRTH = 17 KB	86 KB FCRTL = 21 KB FCRTH = 36 KB
No	No	32 KB	64 KB FCRTL = 7 KB FCRTH = 37 KB	63 KB FCRTL = 6 KB FCRTH = 36 KB
No	Yes	22 KB	57 KB FCRTL = 7 KB FCRTH = 12 KB	91 KB FCRTL = 21 KB FCRTH = 31 KB
64 KB	No	25 KB	54 KB FCRTL = 7 KB FCRTH = 29 KB	53 KB FCRTL = 6 KB FCRTH = 29 KB
64 KB	Yes	19 KB	44 KB FCRTL = 6 KB FCRTH = 9 KB	78 KB FCRTL = 20 KB FCRTH = 28 KB
64 KB	No	24 KB	56 KB FCRTL = 7 KB FCRTH = 29 KB	55 KB FCRTL = 6 KB FCRTH = 28 KB
64 KB	Yes	14 KB	49 KB FCRTL = 3 KB FCRTH = 4 KB	83 KB FCRTL = 18 KB FCRTH = 23 KB
128 KB	No	17 KB	46 KB FCRTL = 7 KB FCRTH = 21 KB	45 KB FCRTL = 6 KB FCRTH = 21 KB
128 KB	No	16 KB	48 KB FCRTL = 7 KB FCRTH = 21 KB	47 KB FCRTL = 6 KB FCRTH = 20 KB

Notes:

1. In some of the cases above, it has been necessary to get compromised on the rules for hysteresis and FCRTL to fit the size available for Rx packet buffer.
2. In some other cases, after having applied all the rules there was an exceeding available Rx packet buffer left which has been used to extend the hysteresis budgets.
3. In all cases, FCRTH rule has been applied as is, since compromising on it is not allowed and extending it provides no performance benefits.

3.6.5.4 Link FC in DCB Mode

When operating in DCB mode, PFC is the preferred method of getting the best use of the link for all TCs. When connecting to switches that do not support (or enable) PFC, the E610 can also throttle the traffic according to incoming link FC notifications. Following is the required device setting and functionality.

- The E610 should be set to legacy link FC by setting `MFLCN.RFCE`.
- Receive XOFF pauses transmission in all TCs.
- Crossing the Rx buffer high threshold on any TC generates XOFF transmission. Each TC can have its own threshold configured by the `FCRTH[n]` registers.
- Crossing the Rx buffer Low threshold on any TC generates XON transmission. This behavior is undesired. Therefore, software should not enable XON in this mode by clearing `FCRTL[n].XONE` bits in all TC.
- The `FCTTV` of all TCs must be set to the same value.

3.6.6 Inter Packet Gap (IPG) Control and Pacing

The E610 supports transmission pacing by extending the IPG (the gap between consecutive packets). The pacing mode enables the average data rate to be slowed in systems that cannot support the full link rate (10 GbE, 1 GbE or 100 Mb/s). As listed in [Table 3-78](#), the pacing modes work by stretching the IPG in proportion to the data sent. In this case the data sent is measured from the end of preamble to the last byte of the packet. No allowance is made for the preamble or default IPG when using pacing mode.

Example 1:

Consider a 64-byte frame. To achieve a 1 GbE data rate when link rate is 10 GbE and packet length is 64 bytes (16 DWords), add an additional IPG of 144 DWords (nine times the packet size to reach 1 GbE). When added to the default IPG gives an IPG of 147 DWords.

Example 2:

Consider a 65-byte frame. To achieve a 1 GbE data rate when link rate is 10 GbE and packet length is 65 bytes (17 DWords when rounded up) add an additional IPG of 153 DWords (nine times the packet duration in DWords). When added to the default IPG gives an IPG of 156 DWords. Note that in this case, where the packet length counted in DWords is not an integer, count any fraction of a DWord as an entire DWord for computing the additional IPG.

[Table 3-78](#) lists the pacing configurations supported by the E610 at link rates of 10 GbE. When operating at lower link speeds the pacing speed is proportional to the link speed.

Table 3-78. Pacing Speeds at 10 GbE Link Speed

Pacing Speeds (Gb/s)	Delay Inserted into IPG	Register Value
10 (LAN)	None	0000b
9.294196 (WAN)	1 byte for 13 transmitted	1111b
9.0	1 DWord for 9 transmitted	1001b
8.0	1 DWord for 4 transmitted	1000b
7.0	3 DWords for 7 transmitted	0111b
6.0	2 DWords for 3 transmitted	0110b

Table 3-78. Pacing Speeds at 10 GbE Link Speed [continued]

Pacing Speeds (Gb/s)	Delay Inserted into IPG	Register Value
5.0	1 DWords for 1 transmitted	0101b
4.0	3 DWords for 2 transmitted	0100b
3.0	7 DWords for 3 transmitted	0011b
2.0	4 DWords for 1 transmitted	0010b
1.0	9 DWords for 1 transmitted	0001b
10	None	Default

Pacing is configured in the *PACE* field of the Pause and Pace (PAP) register.

3.7 Link Management

3.7.1 Overview

The E610 incorporates a PHY Subsystem with two lanes supporting a wide range of Ethernet Network Interface (ENI) configurations.

The Link Manager refers to a specific firmware with the collection of features and functionality of managing Ethernet links implemented within the Link Task and associated DNL Scripts. This includes all link management flows and access to the Link Topology through the defined sideband interfaces.

3.7.1.1 Features

The Link Manager provides the following services:

- Discover the link topology
- Initialize the link topology
- Detect link capabilities
- Establish link
- Monitor link status
- Update LED indicators (if firmware option is chosen)
- Monitor PHY and Pluggable Media temperature sensors
- Service Pluggable Media hot-plug events
- Provides Link Manager and PHY SS logs

For a detailed description of the corresponding internal flows please refer to [Section 3.7.2](#).

3.7.1.2 Components

The Link Manager is built around the following components:

- **Link Management Engine** — Implemented as a periodically scheduled task, scheduled on a regular 100 ms cadence by the Link Manager, it is responsible for orchestrating the Link Topology Flows and Link Configuration Flows.
- **Topology Engine** — Responsible for providing description of the Link Topology and the necessary configuration information for the Link Configuration Flows.
- **Dreadnought Lake (DNL) Engine** — Responsible for executing the scripts implementing the Link Topology Flows, Link Configuration Flows and providing an abstraction layer of the Link Topology components.

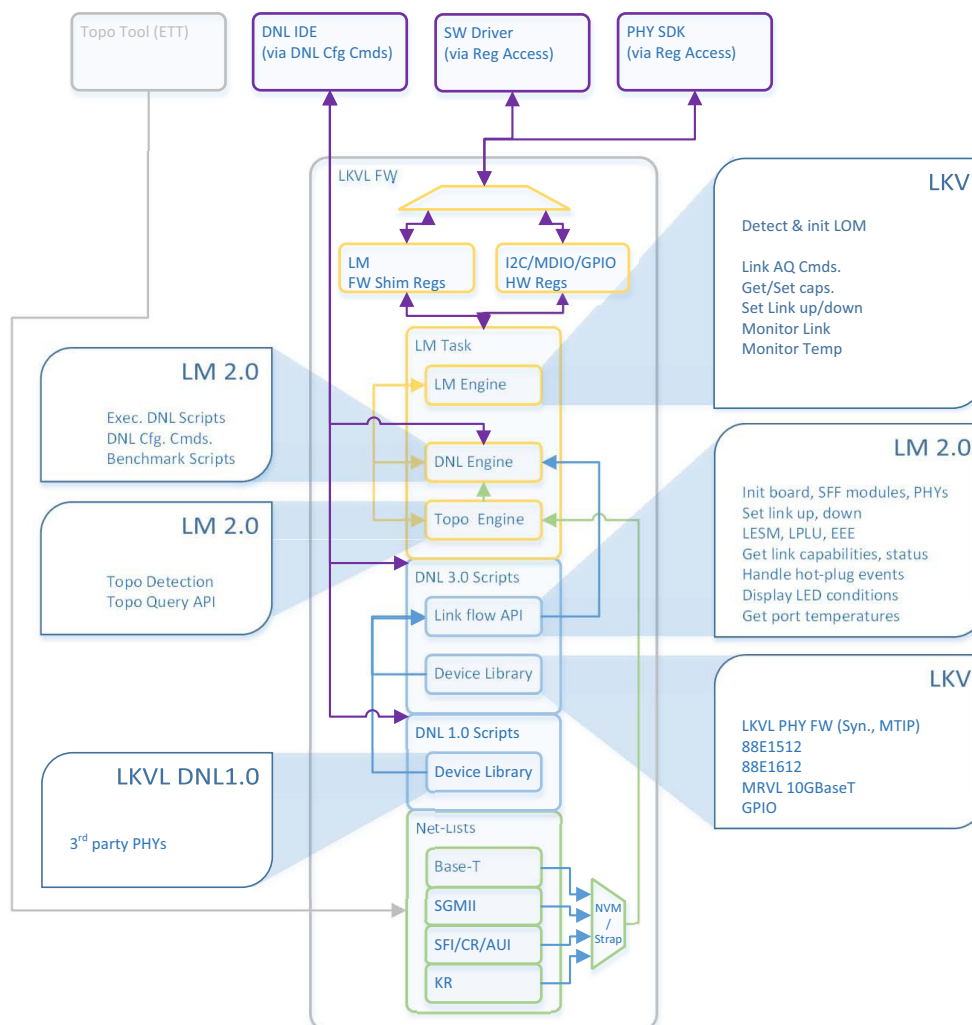


Figure 3-18. Link Manager Components

3.7.1.3 Flows

The flows implemented by the Link Manager are divided into two categories:

- Initialization flows related to the detection and initialization of the link topology and the components within, described in Link Topology Flows.
- Periodic flows related to link establishment and maintenance detailed in Link Configuration Flows.

These flows are implemented by the Link Task calling the initialization functions documented in LM Initialization, Topology Initialization, DNL Engine Initialization, the periodic handler every 100 ms and the interrupt service routines on every interrupt on a GPIO owned by Link Manager.

To expose control over these flows the following set of commands are provided:

- **Link Topology Commands** — Expose access to the components within the link topology.
- **Link Commands** — Expose link configuration and status monitoring.

- **Dreadnought Lake Commands** — Expose debug functionality for the Dreadnought Lake scripting environment.

3.7.2 Link Flows

3.7.2.1 Overview

This section defines the flows used for link establishment including detecting PHY capabilities, link establishment, monitoring and indicating link status and disabling link.

In the E610 the implementation of link management is split between the link task, part of the EMP firmware, and the Dreadnought Lake (DNL) activities executed by the DNL Engine instantiated by the EMP firmware.

Link initialization includes determining the PHY capabilities based on topology netlist information and if applicable the connected media type, configuring the connected pluggable modules, configuring external and integrated PHY devices as well as programming appropriate values to the MAC registers to bring the link up.

The flow diagram in [Figure 3-19](#) is color coded. Tasks with orange outline are performed by the Link Manager. Tasks with blue outline are performed by DNL activities. Tasks with purple outline are performed by the software driver.

The Link Management flows are controlled by commands from manageability or the software driver. [Figure 3-19](#) shows the interaction between these software entities and the per port link management flows implemented by the Link Manager and DNL activities.

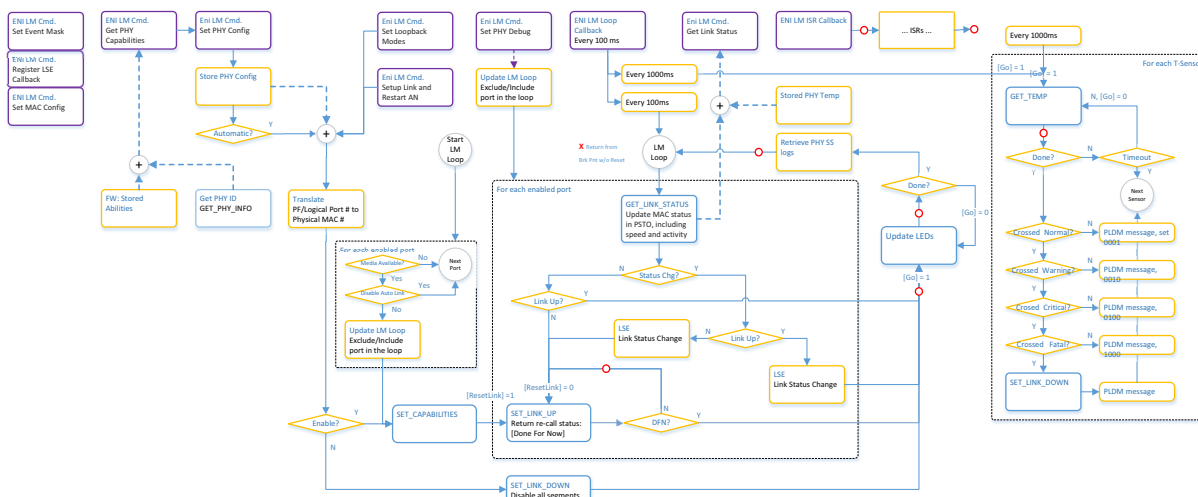


Figure 3-19. Link Configuration Flow

After startup of the Link Manager the flow is initiated by discovering the board topology including both the motherboard and any potential mezzanine cards present in the system. After discovery, each board is initialized. For more detail please refer to [Section 3.7.6, "Topology Flows"](#).

Link management is performed only on the subset of ports which have a medium available or have no method of detecting media presence (i.e. BASE-T or backplane interfaces). The Link Manager detects whether a port needs to be included in the list of actively managed ports based on the discovered PHY abilities returned by the GET_PHY_CAPABILITIES_0 and GET_PHY_CAPABILITIES_1 DNL activities.

For all actively managed ports, the link management may be in three different major states:

- Link bring up, implemented by Link Manager periodically calling the SET_LINK_UP DNL activity until all link segments are configured and link is established.
- Link status monitoring, implemented by Link Manager periodically calling the GET_LINK_STATUS DNL activity to retrieve the current link status.
- Link disable, implemented by Link Manager calling the SET_LINK_DOWN DNL activity to bring link down.

Transitioning between the above states is a function of link status changes, topology changes as well as software driver and manageability link status change requests. The flow is orchestrated by the Link Manager while the above three major tasks are abstracted by DNL scripts. The Link Manager ensures that each actively managed port is serviced on a regular cadence of 100 ms.

The Link Manager decides when to initiate link to be established on a given port. Once link on that port has been established Link Manager moves on to monitoring the link status of the port. In the event link goes down, if the medium is still present, the Link Manager re-initiates link establishment. Based on manageability or software driver request the Link Manager may also disable link on a port in which case link management on the port is suspended.

3.7.2.2 PHY Capability Detection

For each port, the Link Manager needs to detect the PHY capabilities. These are determined by calling the GET_PHY_CAPABILITIES_0 and GET_PHY_CAPABILITIES_1 DNL activities, which returns the list of possible PMD type, speed, and medium combinations also referred to as PHY types.

This information is exposed to the control plane software through the Get PHY Capabilities configuration command. The returned capabilities describe the parameters of the outermost link segment. All potential internal link segments are treated as transparent at this level.

The PHY abilities are detected after the board initialization is complete and in case of pluggable interfaces any time after a module is inserted. The capabilities are identified by intersecting the PHY capabilities configured by the topology netlist, the fuse configuration, and the capabilities allowed for by the attached medium. When the topology does not include pluggable media the PHY types are determined purely based on topology netlist and fuse configuration information.

If module qualification is enabled the capabilities of the present pluggable media are evaluated only after verifying the module is part of the qualified module list.

To complete the intersection of all available capabilities of all PHYs in the PHY chain, the GET_PHY_CAPABILITIES_0 and GET_PHY_CAPABILITIES_1 are repeatedly called until they return with the done bit set. In the process, the link topology netlist defined capabilities for the line-side of the outermost PHY are intersected with the capabilities supported by the connected medium when applicable, then the result is matched with the available speeds and modes of the inner segments.

In some cases, the inner link segments may be chosen to run at a higher speed than the line-side interface of the outermost PHY and Link Flow Control may be used for rate matching the two. For example, if the outermost PHY establishes a 5GBASE-T link but is connected to a PHY which does not support any 5G speeds but supports 10G speeds, then all inner PHY segments are configured for 10G, and Link Flow Control may be enabled. The capabilities (PHY types, etc.) available and supported by all PHYs and link segments are then saved in the persistent store (PSTO) space and returned to Link Manager.

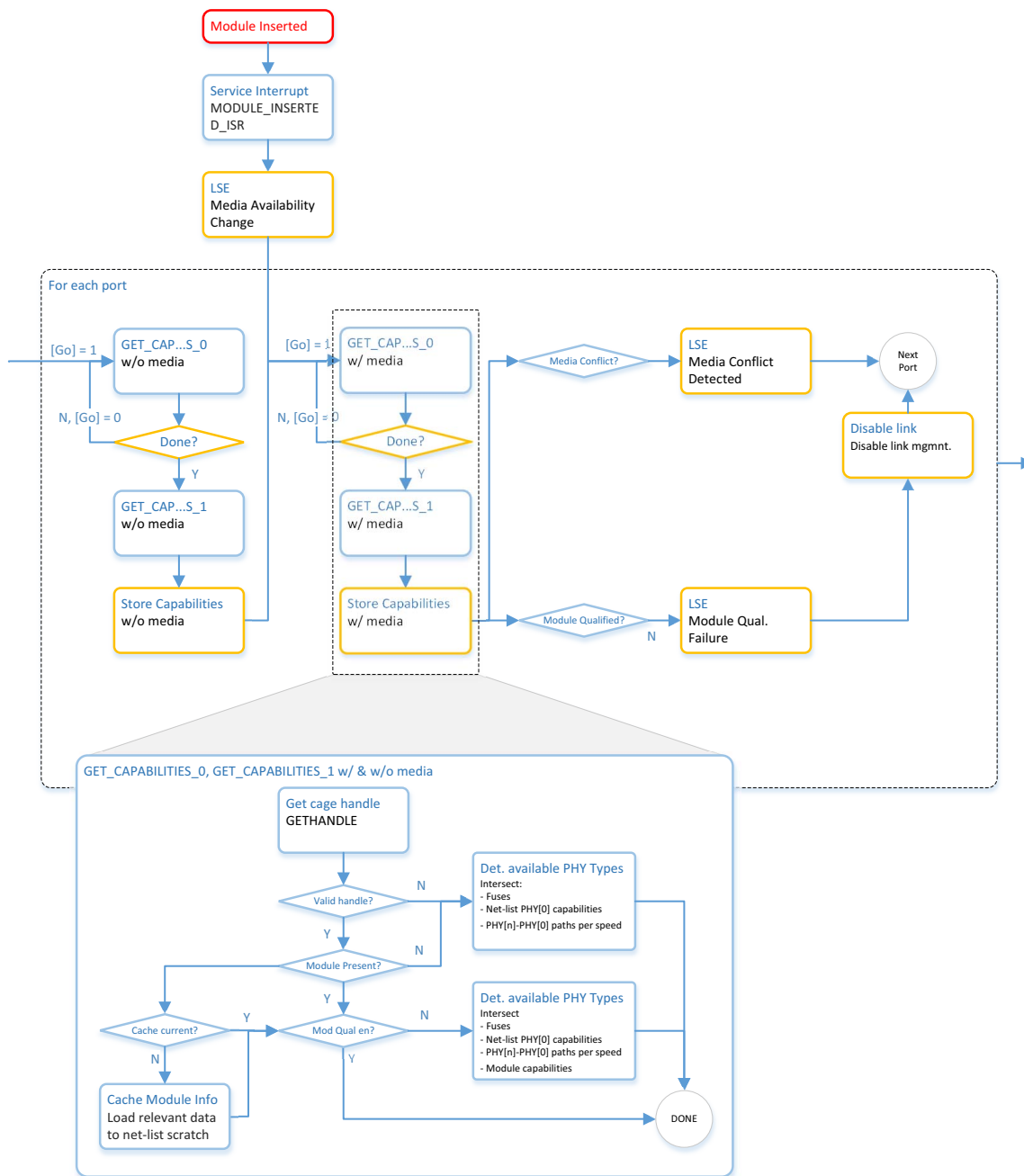


Figure 3-20. PHY Capability Detection Flow

3.7.2.2.1 SFP+, SFP28 PHY Capabilities

To identify the capabilities of SFF-8472 defined SFP+ or SFP28 modules, the following locations are read:

Table 3-79. SFF Defined SFP+, SFP28 Module Capability Registers

I ² C Address	Offset	Register Name
0xA0	0x00	SFF_IDENTIFIER
0xA0	0x06	SFF_1GBE_COMP_CODES_SFP
0xA0	0x03	SFF_10GBE_COMP_CODES_SFP
0xA0	0x24	SFF_EXT_COMP_CODES_SFP
0xA0	0x08	SFF_CABLE_TECHNOLOGY_SFP
0xA0	0x3C	SFF_SFP_CABLE_SPEC_COMP_SFP
0xA0	0x0C	SFF_BR_NOMINAL_SFP
0xA0	0x42	SFF_BR_MAX_SFP
0xA0	0x40	SFF_PWR_CLASS_SFP
0xA0	0x25:0x27	SFF_SFP_OUI
0xA0	0x28:0x37	SFF_SFP_PN_x
0xA0	0x37:3A	SFF_SFP_REV

To identify the capabilities of CMIS defined SFP28 or SFP-DD modules, the following locations are read:

Table 3-80. CMIS Defined SFP28 Module Capability Registers

I ² C Address	Page	Bank	Offset	Register Name
0xA0	N/A	N/A	0x00	SFF_IDENTIFIER
0xA0	N/A	N/A	0x55	CMIS_MODULE.MEDIA_TYPE_ENCODING
0xA0	N/A	N/A	0x56+i*4, i=[0:7]	CMIS_APP[i].HOST_ELECTRICAL_INTERFACE_ID
0xA0	N/A	N/A	0x57+i*4, i=[0:7]	CMIS_APP[i].MODULE_MEDIA_INTERFACE_ID
0xA0	N/A	N/A	0x58+i*4, i=[0:7]	CMIS_APP[i].HOST_MEDIA_LANE_COUNT
0xA0	N/A	N/A	0x59+i*4, i=[0:7]	CMIS_APP[i].HOST_LANE_ASSIGNMENT_OPTIONS
0xA0	0x01	N/A	0xB0+i, i=[0:15]	CMIS_APP[i].MEDIA_LANE_ASSIGNMENT_OPTIONS
0xA0	0x01	N/A	0xDF+(i-8)*4, i=[8:15]	CMIS_APP[i].HOST_ELECTRICAL_INTERFACE_ID
0xA0	0x01	N/A	0xE0+(i-8)*4, i=[8:15]	CMIS_APP[i].MODULE_MEDIA_INTERFACE_ID
0xA0	0x01	N/A	0xE1+(i-8)*4, i=[8:15]	CMIS_APP[i].HOST_MEDIA_LANE_COUNT
0xA0	0x01	N/A	0xE2+(i-8)*4, i=[8:15]	CMIS_APP[i].HOST_LANE_ASSIGNMENT_OPTIONS
0xA0	0x01	N/A	0x8C	CMIS_PWR_CLASS
0xA0	0x01	N/A	0xC9	CMIS_MAX_PWR
0xA0	0x01	N/A	0x91:0x93	CMIS_VENDOR_OUI
0xA0	0x01	N/A	0x94:0xA3	CMIS_VENDOR_PN
0xA0	0x01	N/A	0xA4:0xA5	CMIS_VENDOR_REV

The module is considered not supported in the following cases:

- If the SFF_IDENTIFIER value is not 0x03 or 0x1A.
- The media is identified as 10gbe_comp_10gbase_er.
- The media is identified as 10gbe_comp_10gbase_lrm.

The following logic based on module EEPROM data is used to enable the listed PHY capabilities:

```

if(1gbe_comp_1000base_sx)
    add_media_phy_cap(1000BASE_SX)
if(1gbe_comp_1000base_lx)
    add_media_phy_cap(1000BASE_LX)
if(1gbe_comp_1000base_t)
    add_media_phy_cap(1000BASE_T, 100BASE_TX)
if(ext_comp_5gbase_t_cap)
    add_media_phy_cap(5GBASE_T)
if(ext_comp_2p5gbase_t_cap)
    add_media_phy_cap(2P5GBASE_T)
if(10gbe_comp_10gbase_er)
    add_media_phy_cap(); set(unsupported_module)
if(10gbe_comp_10gbase_lrm)
    add_media_phy_cap(); set(unsupported_module)
if(10gbe_comp_10gbase_sr)
    add_media_phy_cap(10GBASE_SR)
if(10gbe_comp_10gbase_lr)
    add_media_phy_cap(10GBASE_LR)
if(ext_comp_10gbase_t_cap)
    add_media_phy_cap(10GBASE_T)
if(ext_comp_10gbase_t_short_cap)
    add_media_phy_cap(10GBASE_T)
if(cable_tech_sfp_plus_passive)
    add_media_phy_cap(10G_SFI_DA)
if(cable_tech_sfp_plus_active)
    add_media_phy_cap(10G_SFI_AOC_ACC)

```

Any of the above PHY types may be masked by the topology netlist or fuse configuration and further restricted via the Set PHY Config configuration command before they are used for link establishment.

3.7.2.2.2 FEC PHY Capabilities

This section describes how the Link Manager controls the FEC abilities and FEC advertisement of the PHYs during auto-negotiation.

- **Fire code Forward Error Correction (KR-FEC), also referenced as shortened cyclic code (2112,2080)** — Defined in IEEE 802.3, Clause 74. This clause defines the use of the Fire code FEC for lane speeds of 10G in two PHY types of single lane (10GBASE-KR).

Use of most FEC modes can be controlled by the associated *Ability* and *Request* bits in the link configuration commands. In case of some PHY types where use of FEC is mandatory and not optional, the FEC input field can be ignored.

For auto-negotiated link modes, the advertisement of a particular FEC capability can be controlled by the software driver by setting the corresponding *Ability* bit. Similarly, setting the associated *Request* bits results in requesting the relevant FEC mode during auto-negotiation. The actual FEC configuration applied is a result of the priority resolution calculated based on the local and the link partner's advertisement.

For non-negotiated link modes, the *Ability* bits are ignored and the *Request* bits are treated as enable bits of their corresponding FEC mode.

3.7.2.3 Link Up Flow

For each port, the link path may consist of several link segments. Link establishment is done by configuring the outermost segment first, followed by configuring all of the inner segments with a matching/supporting speed, as illustrated in Figure 3-21.

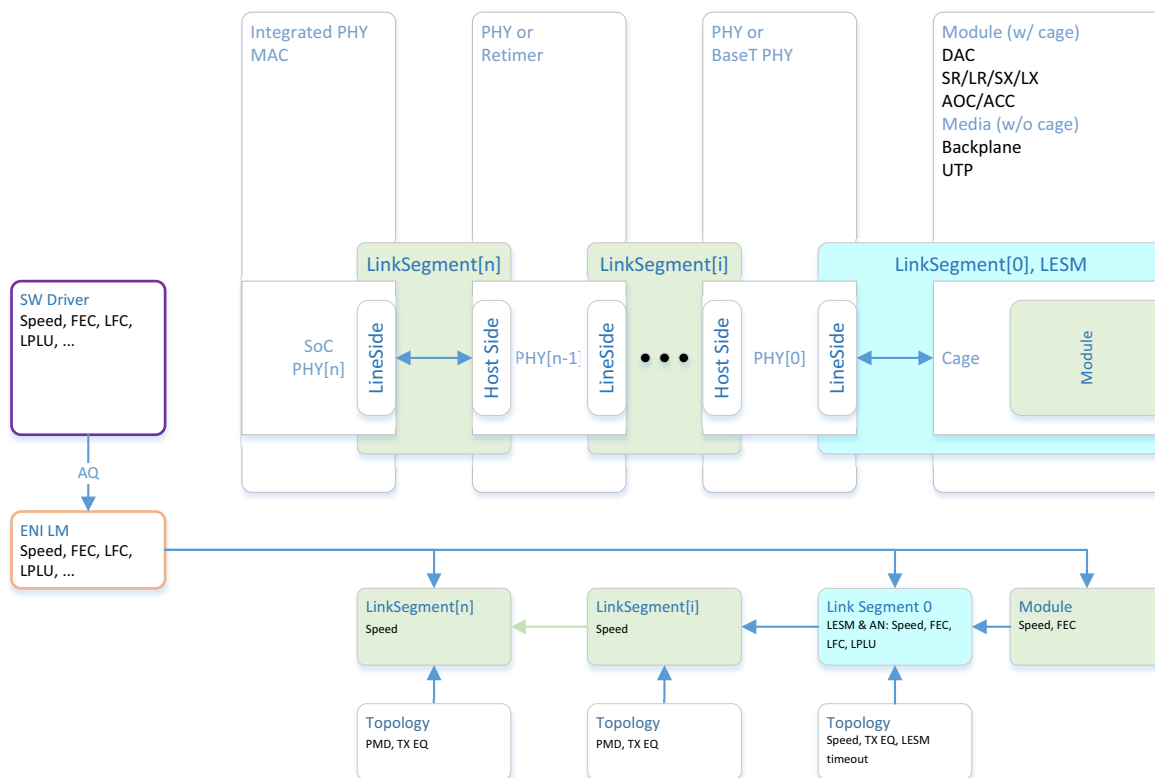


Figure 3-21. Link Segments of a Single Port

The PHY that is capable of auto-negotiation or has an enabled PCS, usually the outermost PHY in the path, is considered the controlling PHY. The link speed and mode of all other PHYs in the chain are determined based on the convergence of this PHY with the Link Partner. Once link is up on the controlling PHY, all of the inner segments are configured accordingly.

Once the Link Manager has completed the link topology initialization flows and determined the available capabilities, depending on the value of the Disable Auto Link on Startup defined by Link Options 1 - 0x0005, it begins the link establishment process. This process can be restarted through the Set PHY Config configuration command with the *Enable Link* and *Enable Automatic Link Update* bits set, or through the Setup Link and Restart Auto-Negotiation configuration command. Anytime this process is started, either out of reset, through a configuration command or insertion of a pluggable module, the Link Manager proceeds to call the SET_LINK_UP DNL activity with the *RESETLINK* bit asserted. This reinitializes the state machines implemented within the activity and configures the port based on the SET_LINK_UP input parameters, the topology netlist, and the PHY Types enabled through the SET_PHY_CAPABILITIES DNL activity.

Once the SET_LINK_UP activity has been initialized with the *RESETLINK* bit, Link Manager continues orchestration of link bring-up following the loop shown in Figure 3-22. Unless the process needs to be restarted, all subsequent calls to SET_LINK_UP are placed with the *RESETLINK* bit cleared. The input arguments are sampled only when *RESETLINK* is set and ignored otherwise.

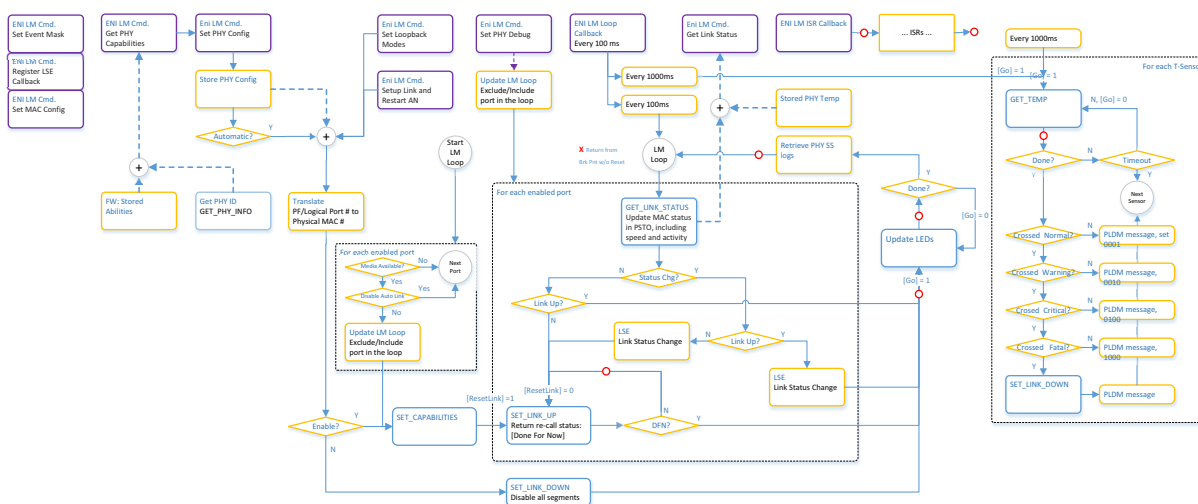


Figure 3-22. Link Configuration Flow

The repeated calls to SET_LINK_UP facilitate the state machine shown in Figure 3-23.

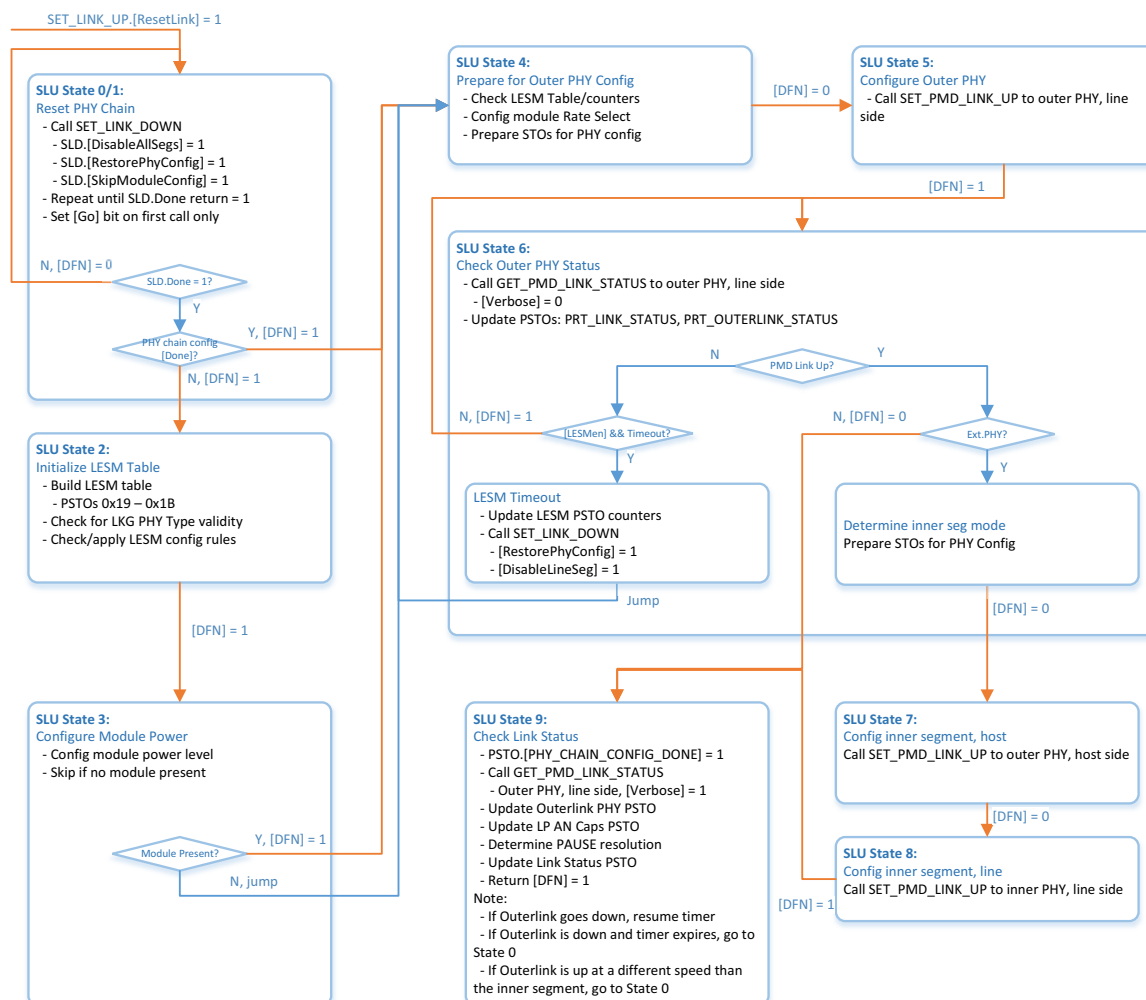


Figure 3-23. SET_LINK_UP State Machine

The state machine implemented within the SET_LINK_UP activity is managing the phases of the link up process. Every time SET_LINK_UP completes its current iteration, it indicates to the Link Manager whether or not it needs to be called again immediately through the *DONEFORNOW* (DFN) bit. SET_LINK_UP only requests to be called again immediately during time sensitive periods of the link up process which may impact time to link or PHY interoperability.

Embedded within this flow SET_LINK_UP implements a Link Establishment State Machine (LESM), responsible for iterating over multiple different PHY types to establish link with a link partner with unknown or non-standard PHY configuration. LESM is enabled through the link topology netlist and/or through configuration commands. If it is not, the outer PHY is configured for a single PHY type (when auto-negotiating, multiple auto-negotiation enabled PHY types can be advertised), and when that link is established, the rest of the PHY chain is configured accordingly.

The standard link up process uses up to ten states. Some states may be skipped depending on the system configuration.

- **State 0** — Used to restart the LESM process and restore the PHY chain to its initial configuration, this state is entered in two scenarios: outer link was up but is lost without receiving a *RESETLINK*, or outer link drops and is reestablished without a timeout but at a different speed after an inner segment was already configured according to the original outer link speed. In this case the LESM tables do not need to be reinitialized, so the state machine continues in State 4.
- **State 1** — Entered any time *RESETLINK* is asserted. This is expected to occur the first time the port is enabled, anytime that a reset is requested via configuration commands, and anytime the media changes. The PHY chain is restored to its initial configuration and the LESM process is reinitialized.
- **State 2** — The the configuration is validated against the LESM rules and the LESM tables are reinitialized. See [Section 3.7.2.3.1](#) for more detail.
- **State 3** — The state machine checks for any pluggable module requiring power level configuration. If the module is an optical module the laser is enabled. If there is no SFP+/SFP28 cage present the state machine proceeds to State 4.
- **State 4** — This state determines how to configure the outermost PHY based on the LESM table. This decision is made each time a new PHY type is to be configured, such as after an LESM timeout. If required, this is when configuration of the Rate Select of an optics module takes place.
- **State 5** — Executed immediately after the previous, the outermost PHY is configured.
- **State 6** — Used to monitor the outermost PHY link status when there is an external PHY, to determine if the proper PHY type/LESM state has been chosen. The result is used to determine if the rest of the PHY chain should be configured, or change PHY types if there is an LESM timeout.
- **State 7** — Configures the host-side of the outer PHY on internal link segments.
- **State 8** — Configures the line-side of the inner PHY on internal link segments.
- **State 9** — The final state of the flow gathers the detailed outer PHY link info, such as link partner capabilities as applicable and perform link flow control PAUSE resolution. This step is repeated any time that GET_LINK_STATUS recognizes a MAC link status change, to make sure that the reported link parameters accurately reflect the change in status.

3.7.2.3.1 Link Establishment State Machine (LESM)

The Link Establishment State Machine, operating independently for each port, is responsible for determining the valid PHY type and speed combinations for a given medium and then cycling through the enabled PHY types, configuring the controlling PHY accordingly until PHY link has been achieved with the connected link partner. The key features provided are:

- Link enablement of available PHY Types (all AN and up to AU1).
- Automatic Link establishment with partners of unknown configuration.
- Support of multiple FEC modes on AU1 links.
- Potential Time to Link optimization utilizing the Last Known Good (LKG) LESM state.

Embedded within the SET_LINK_UP state machine the flow is facilitated by the Link Manager repeatedly calling GET_LINK_STATUS and SET_LINK_UP activities as described in [Section 3.7.2.3](#).

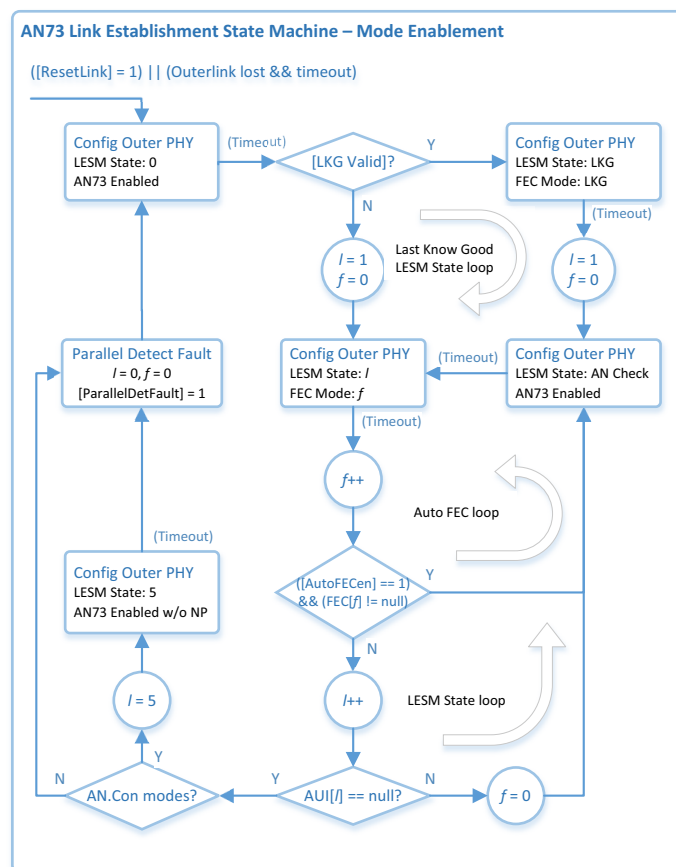


Figure 3-24. LESM Flow

- **LESM State** — The current PHY type configuration, indexed among all allowed and enabled PHY types by the variable "I". LESM State I = 0 refers to auto-negotiation clause 73 configuration.
- **FEC Mode** — The current FEC configuration, indexed among all allowed and enabled FEC configurations of the current PHY type by the variable "f". FEC modes include but are not limited to IEEE clause 74 Firecode FEC, clause 91 Reed Solomon FEC, clause 108 Reed Solomon FEC, Consortium Low Latency Reed Solomon FEC, or No FEC (disabled).
- **Outer PHY** — The line-side of the outermost PHY. If the device uses an integrated PHY, it refers to the host PHY configuration. In case there is a chain of PHYs used to connect the system to the link partner, it is the line-side of the PHY which connects to the link partner.
- **LKG** — Last Known Good LESM state, equivalent to a previous link configuration which successfully achieved link without using auto-negotiation.
- **LKG Valid** — Boolean variable indicating if there is a valid Last Known Good configuration.
- **AN Check** — A state where the PHY configuration is equivalent to LESM State 0 auto-negotiation, performed for a very brief amount of time in order to determine if an auto-negotiation link partner is present.
- **Auto FEC Enable** — Boolean variable configuring whether or not the state machine should enable all available FEC modes, or just the highest priority FEC mode (f = 0).

- **Con modes** — Ethernet Consortium Modes. If any Ethernet Consortium modes are enabled through auto-negotiation, they are enabled using auto-negotiation Next Pages.
- **LESM State Max** — LESM state "I" = max is an equivalent configuration to LESM State 0, except that all configurations requiring auto-negotiation Next Pages are disabled. This allows interoperability with non-compliant devices which do not support auto-negotiation Next Page exchange.
- **Parallel Detect Fault** — Indication that all LESM states have been attempted without successfully establishing link. In the event of a Parallel Detect fault, the LESM State 0 timer must be increased to a value that is greater than the sum of all of the other LESM States in order to ensure that a deadlock does not occur once a properly configured link partner is present. This indication is reset to 0 anytime that link is established.
- **Timeout** — Expiration of a timer which starts after the PHY configuration is complete. The timeout value for each LESM state may be configurable via netlist properties to allow for different receiver convergence time requirements per PHY type or speed.

Certain PHY configurations are not possible and cannot be resolved to an allowed configuration, and therefore results in an error if they are requested. Other configurations may result in certain requested PHY types not being enabled, in order to ensure a robust link configuration. If this occurs, SET_LINK_UP disables the modes internal to the link establishment process.

Table 3-81 shows the LESM Configuration rules and how they are enforced.

Table 3-81. LESM Configuration Rules

LESM Enabled	Auto-Neg	Number of AUI PHY Types	Link Configuration Result
Don't care	None	0	Error due to no PHY types set.
Don't care	Clause 73 Clause 28	Don't care	Error due to conflicting AN configuration.
Don't care	Clause 28	0	Configuration unchanged.
Don't care	Clause 28	1+	Error due to conflicting PHY configuration. For example, a mix of BASE-T and AUI PHY types configured.
Don't care	Clause 37	Don't care	Clause 37 auto-negotiation treated as AUI mode for LESM.
No	Clause 73	0	Configuration unchanged.
No	Clause 73	1+	Only clause 73 PHY types enabled.
No	None	1	Configuration unchanged.
No	None	2+	Only highest priority PHY type enabled.
Yes	Clause 73 Clause 28	0	Configuration unchanged.
Yes	Clause 73	1+, no active media	Configuration unchanged.
Yes	Clause 73	1+, active media	Error due to conflicting configuration.
Yes	None	1	Configuration unchanged.
Yes	None	2+, no active media	Only highest priority PHY type enabled.
Yes	None	2+ active media ¹	Configuration unchanged.

1. Multiple AUI PHY types allowed without AN and enabled through LESM if module is active media.

Additionally if a PHY type is requested that requires auto-negotiation, and the corresponding *AN ENABLE* bit is not set in the netlist, *SET_LINK_UP* returns an error.

Given a set of link modes to attempt, including both auto-negotiated and non-auto-negotiated link modes, first the order of link modes to configure is determined. The order is derived based on their appearance in the Extended PHY Capabilities data structure in descending bit order and FEC encoding gain (highest to lowest). The maximum time allotted for each type of link mode attempted is configurable, however in certain cases, for the link mode with highest priority, it may automatically be overridden by the LESM state machine.

When the list of link modes to attempt contains auto-negotiated link modes, the algorithm prioritizes these over the rest of non-auto-negotiated AUI link modes. As a result, first auto-negotiated link modes are attempted followed by attempting AUI link modes in decreasing order of priority determined based on their appearance in the Extended PHY Capabilities data structure in descending bit order. Each link mode may support multiple FEC configurations, which are attempted sequentially as needed.

To optimize time to link and to eliminate corner cases preventing identically configured devices connected back to back from establishing a link, after each attempted AUI mode the auto-negotiated link modes are attempted for a short period of time, long enough to determine if there is a connected link partner transmitting auto-negotiation base pages. If the PHY reports base pages are being received, the device stays in this state and bring up link through auto-negotiation.

This behavior leads to a reduced time to link with all freshly connected or configured auto-negotiation enabled link partners. This feature also helps reduce interoperability issues due to receiver convergence failure in the link partner by enforcing loss of signal between attempting different FEC modes on the same AUI link mode.

If all link modes were attempted and link was not achieved, to accommodate clause 73 auto-negotiation capable link partners which do not support next pages, link modes requiring the use of next pages are removed from the list of modes advertised through auto-negotiation. If this does not result in link up, the removed link modes are advertised in the next attempt at auto-negotiated link modes.

Once all configurations have been attempted the state machine returns to the highest priority link mode and repeats the above described flow. This condition is referred to in the below example as a parallel detect fault, and is cleared any time that link is successfully established. If configured to do so in a parallel detect fault condition, the algorithm extends the timeout value for the highest priority link mode to be the sum of the time allowed for all enabled link modes. This behavior eliminates the risk of corner cases preventing identically configured devices connected back to back from establishing a link.

When the list of link modes to attempt does not contain auto-negotiated link modes, the algorithm attempts the requested AUI modes in decreasing order of priority based on their appearance in the Extended PHY Capabilities data structure. To eliminate corner cases preventing identically configured devices connected back to back from establishing a link, after unsuccessfully attempting all link modes, in case of a parallel detect fault condition, the timeout of the highest priority link mode is extended to be the sum of the time allowed for all enabled link modes.

This mechanism for establishing link can also support media where multiple link modes are supported but none of them are supported by clause 73 auto-negotiation, such as the case of multi-speed optical modules.

To speed up link establishment, the algorithm can alter the above described order of priority for the non-negotiated link modes based on the last known good configuration. When link is attempted for the first time after a link down event, if the last successfully used link mode was an AUI link mode, then after attempting potential auto-negotiated link modes the last known good AUI link mode is attempted first. If unsuccessful, the configured AUI link modes are attempted in decreasing order of priority determined based on their appearance in the Extended PHY Capabilities data structure. This provides improved efficiency in reducing time to link with all non-auto-negotiating link partners.

If the Ethernet device being configured is capable of providing a reliable signal detection indication, the state machine waits for a received signal to be detected before beginning to cycle through the configured link modes. This provides the fastest time to link possible when connected with an auto-negotiation enabled link partner.

This algorithm can be used to configure the Ethernet device for both standard and non-standard link modes, so that interoperability amongst all possibly connected devices is maximized.

The following sections cover the options used to initialize the LESM tables depending on the detected medium. The LESM options are chosen based on the link topology netlist and the GET_PHY_CAPABILITIES_0 DNL activity however they can be further restricted by configuration commands. The following sections represent the default expected configurations for the connected medium, assuming that the netlist for each PHY in the path supports all of the outermost link speeds for the given medium. The option numbers also represents the LESM priority order which the mode is attempted. If multiple FEC types are supported by default on an AUI PHY type and *AUTOFECEEN* is not asserted, only the highest priority FEC type enabled by the netlist and software is attempted in the LESM flow. FEC modes in the table are listed in their priority order. For AN PHY types, FEC is resolved based on IEEE auto-negotiation.

3.7.2.3.2 LESM Options for SFP BASE-T

If the enabled PHY types are 1000BASE-T and 100BASE-TX and the port is routed to an SFP or SFP+ cage, the LESM table is configured to auto-negotiation, with both of those BASE-T modes enabled. This is only possible with an SFP to BASE-T module:

Table 3-82. LESM Options for SFP BASE-T

Option	Auto-Neg.	Speed	PMD Type	Default FEC Mode	Auto FEC Modes
0	Clause 28	100M, 1G	1000BASE-T, 100BASE-TX	No FEC	N/A

3.7.2.3.3 LESM Options for SFP+ Optical

If the outermost PHY is connected to an SFP+ cage, the connected module capabilities are read. Only the speed supported according to the module EEPROM is enabled. If the module supports rate select, then multiple LESM options are enabled, and the LESM table is configured with up to the following two options:

Table 3-83. LESM Options for SFP+ Optical

Option	Auto-Neg	Speed	PMD Type	Default FEC Mode	Auto FEC Modes
1	N/A	10G	10GBASE-SR, 10GBASE-LR	No FEC	No FEC
2	N/A	1G	1000BASE-SR, 1000BASE-LR	No FEC	No FEC

3.7.2.3.4 LESM Options for SFP+ DA

If the connected medium is a 10G-SFI-DA cable, the LESM table is configured with the following option:

Table 3-84. LESM Options for SFP+ DA

Option	Auto-Neg	Speed	PMD Type	Default FEC Mode	Auto FEC Modes
1	N/A	10G	10G-SFI-DA	No FEC	No FEC

3.7.2.3.5 LESM Options for SFP28 DA

If the connected medium is a SFP28 DA cable, the LESM table is configured with the following options. These options may be restricted further based on the cable EEPROM or cable length, since not all FEC modes are supported:

Table 3-85. LESM Options for SFP28 DA

Option	Auto-Neg.	Speed	PMD Type	Default FEC Mode	Auto FEC Modes
2	N/A	10G	10G-SFI-C2C	No FEC	No FEC
3	N/A	1G	1G SGMII	No FEC	No FEC

3.7.2.3.6 LESM Options for Backplane

If the connected medium is a backplane, then there is no module to restrict the modes and LESM is based solely off what is set in the netlist. In backplane cases, it is strongly recommended to disable LESM and/or only use clause 73 auto-negotiation PHY Types, since this is the spec defined method of establishing a backplane link. In case AUI/C2C modes are set in the netlist and LESM is enabled, then the four highest priority modes which are set are used for the LESM flow. The potentially allowed AUI backplane modes are as follows:

Table 3-86. LESM Options for Backplane

Option	Auto-Neg.	Speed	PMD Type	Default FEC Mode	Auto FEC Modes
0	Clause 73	10G 5G 2.5G 1G	10GBASE-KR 5GBASE-KR 2.5GBASE-KX 1000BASE-KX	KR/No FEC No FEC No FEC No FEC	N/A
2	N/A	10G	10G-SFI-C2C	No FEC	No FEC
3	N/A	1G	1G-SGMII	No FEC	No FEC

3.7.2.3.7 LESM Options for All Other Cases

These are the expected mediums to be used in a typical system. Most other PHY types directly map to a single PMD type and speed, and therefore do not change modes during the LESM flow.

3.7.2.3.8 Pluggable Module Power Mode Configuration

This section describes the flow executed by the Link Manager to control the power mode of pluggable modules compliant with the SFF or CMIS specifications.

3.7.2.3.9 Low Power Mode

Each time a pluggable module is detected, either at power on or inserted at a later time, or link is disabled by software driver, the module is initialized to a Low Power mode. As a result the module is limited to Power Class 1.

For SFP modules, this is achieved through I²C by clearing bit 0 of offset 0x76.

3.7.2.3.10 High Power Mode

Since certain modes of operation may require the module to consume more power, before link can be established, the Link Manager has to enable the module to enter High Power mode.

To enable this mode two conditions must be met:

- High Power Mode must be allowed by the netlist and software driver configuration. See Low Power Mode defined in Link Options 1 - 0x0005 and Low Power Mode defined in Set PHY Config Command Inputs.
- The module's reported maximum power consumption must be lower than the Module Max Power allowed by the netlist for the specific cage. See Module Max Power Hi defined in Node Address - 0x0002 and Module Max Power Lo defined in Node Part Number and Node Options - 0x0003.

When the above conditions are not met the module is configured to a Low Power mode as described above, otherwise the module is configured to enter High Power mode as follows:

- For SFP modules, this is achieved through I²C by setting bit 0 of offset 0x76.

3.7.2.3.11 Module Power Class

The maximum power consumption of a particular module is determined based on the reported Power Class as defined by the relevant SFF/CMIS specification.

Table 3-87. Module Power Classes

Power Class	Max Power SFP (W) [SFF-8431]
1	1
2	1.5
3	2

3.7.2.3.12 Loopbacks

The Link Manager allows for Local or Remote loopback to be enabled on any PHY within the chain using the Set PHY Loopback configuration command. The loopbacks can be enabled at the PCS or the PMD level, however only the loopback is configured on the line-side of the PHY.



Figure 3-25. Loopbacks

Prior to requesting a loopback first the PHY type must be configured through the Set PHY Config configuration command. Only a single PHY type is supported and requesting loopback with multiple PHY types enabled results in an Link Configuration Error.

3.7.2.3.13 Link Flow Control (LFC)

Link Flow Control is controlled by Link Manager and may be enabled in the following scenarios:

- LFC requested through auto-negotiation with the link partner.
- LFC requested for a non-negotiated AUI PHY type.
- The outer PHY speed is lower than the corresponding MAC configuration requiring LFC on the internal link segments, used for rate adaptation.

The above situations are detected at different times in the link up flow. Figure 3-26 outlines the involved flow:

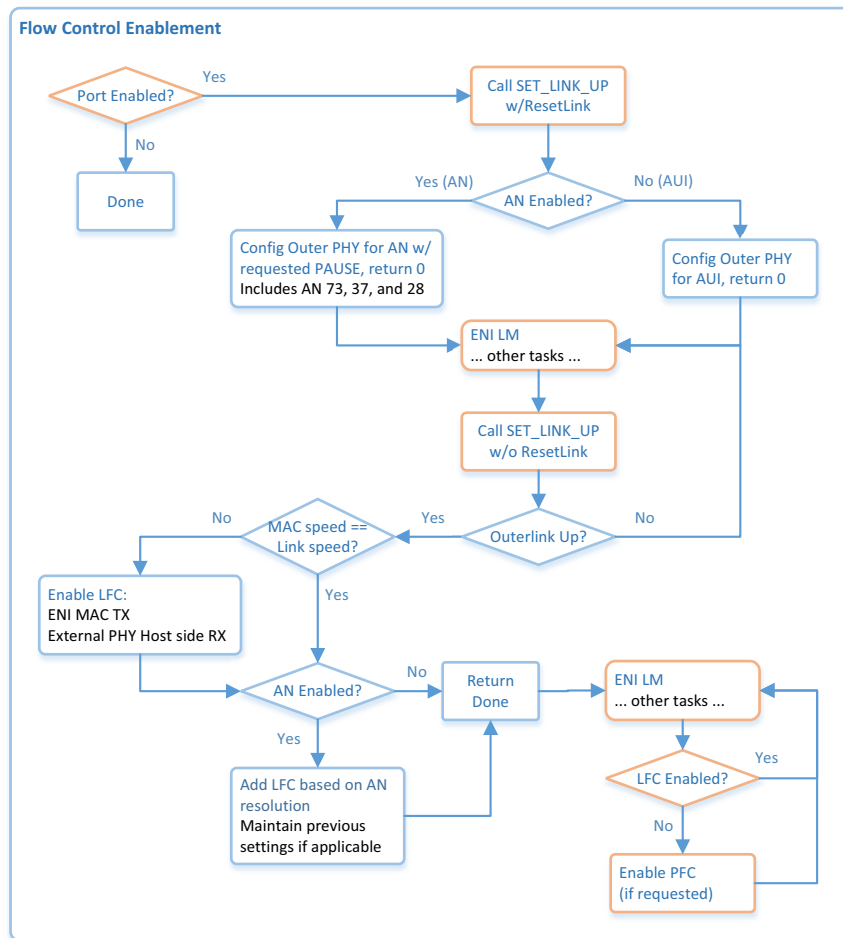


Figure 3-26. Link Flow Control Flow

When auto-negotiating with the link partner, the PHY is configured to advertise PAUSE capabilities using two bits defined within the auto-negotiation base page (see IEEE 802.3 Annex 31B for more details):

- **PAUSE** — Device is capable of providing symmetric PAUSE function.
- **ASM_DIR** — The device support asymmetric PAUSE functionality.

Based on these settings, the devices are able to determine how to configure LFC using IEEE Table 28B-3, which dictates the LFC resolution:

Table 3-88. LFC Priority Resolution Table

Local Device		Link Partner		Local Device Resolution	Link Partner Resolution
PAUSE	ASM_DIR	PAUSE	ASM_DIR		
0	0	Don't care	Don't care	Disable PAUSE Transmit and Receive	Disable PAUSE Transmit and Receive
0	1	0	Don't care	Disable PAUSE Transmit and Receive	Disable PAUSE Transmit and Receive
0	1	1	0	Disable PAUSE Transmit and Receive	Disable PAUSE Transmit and Receive
0	1	1	1	Enable PAUSE Transmit Disable PAUSE Receive	Enable PAUSE Receive Disable PAUSE Transmit
1	0	0	Don't care	Disable PAUSE Transmit and Receive	Disable PAUSE Transmit and Receive
1	Don't care	1	Don't care	Enable PAUSE Transmit and Receive	Enable PAUSE Transmit and Receive
1	1	0	0	Disable PAUSE Transmit and Receive	Disable PAUSE Transmit and Receive
1	1	0	1	Enable PAUSE Receive Disable PAUSE Transmit	Enable PAUSE Transmit Disable PAUSE Receive

The IEEE defines the above PAUSE behaviors as follows:

- **PAUSE Transmit** — Enable device to halt transmitter when requested by link partner.
- **PAUSE Receive** — Enable device to recognize Rx buffer overflow condition and request. link partner to halt data transmission as needed.

3.7.2.3.14 Low Power Linkup (LPLU)

Low Power Link Up (LPLU) is an energy saving feature that may be enabled when the outermost PHY supports BASE-T technologies. It is a way of establishing a link at the lowest possible speed. When LPLU is requested, SET_LINK_UP verifies that the outermost PHY is a BASE-T PHY and then configures and enables the PHY at the lowest allowed link speed. If link is achieved the flow continues as normal, otherwise the results of the Clause 28 auto-negotiation are used to determine the link partner capabilities and the outermost PHY is reconfigured with the lowest common denominator speed supported by both devices.

3.7.2.4 Link Down Flow

To bring link down the Link Manager repeatedly calls the SET_LINK_DOWN DNL activity until it returns with the *DONE* bit set. The first call is placed with the *GO* bit asserted and brings the line-side link on the outermost PHY. The following calls, placed with identical arguments and the *GO* bit cleared facilitate the state machine shown in Figure 3-27:

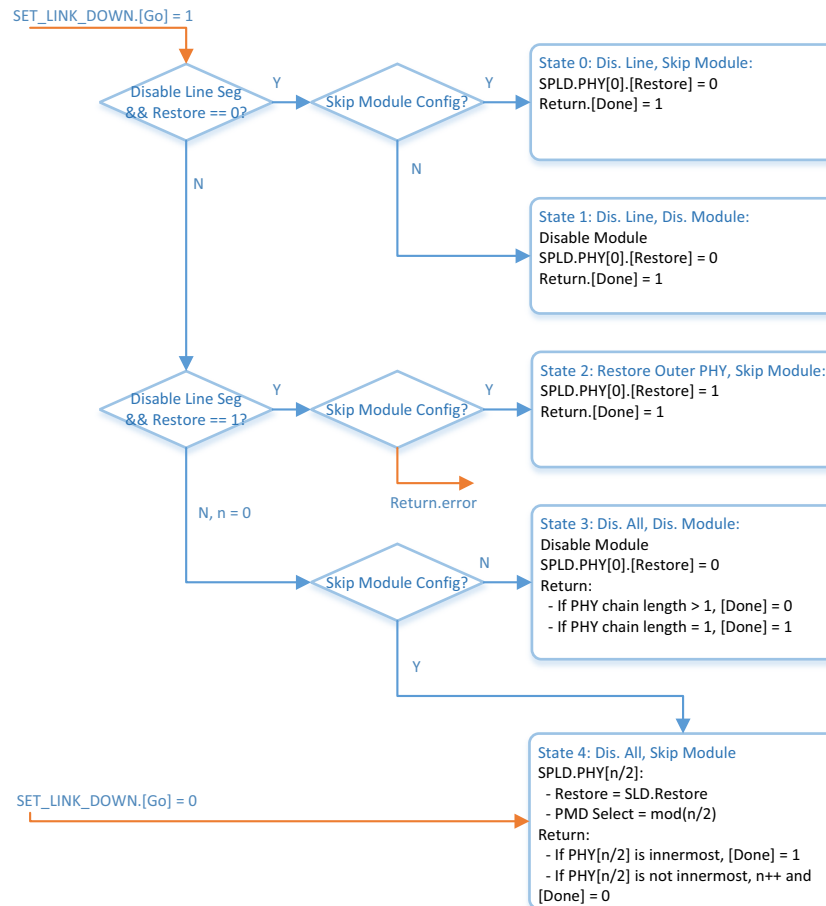


Figure 3-27. Link Down Flow

When disabling link, the flow can either disable the entire PHY chain, or just the line-side of the outermost PHY. The state machine also checks for any pluggable module requiring power level configuration. If the module is an optical module the laser is disabled.

3.7.2.5 Link Status Monitoring

The Link Manager monitors the link status of enabled port by periodically calling the GET_LINK_STATUS DNL activity as shown by the above documented Link Up Flow. The activity is used to monitor the outermost PHY link status as well as the MAC status. The status is saved, used for updating Link Status Indication (LEDs) and available to the control plane software via the Get Link Status configuration command.

Link Manager alternates between calling SET_LINK_UP and GET_LINK_STATUS until GET_LINK_STATUS reports that link is up on a port. Following this, the Link Manager no longer calls SET_LINK_UP on that port, but continues calling GET_LINK_STATUS to monitor for link drop. If a loss of link is detected the Link Manager returns to alternating between calling SET_LINK_UP and GET_LINK_STATUS until link is re-established.

Note: Link events shorter in duration than the frequency of the GET_LINK_STATUS calls are not reported.

3.7.2.6 Link Status Indication (LEDs)

As part of the link management the LEDs defined by the link topology netlist need periodic updates. On a cadence of 200 ms, the Link Management Application triggers an update by repeatedly calling the UPDATE_LEDS DNL activity until it returns with the *DONE* bit set. This state machine implemented within the UPDATE_LEDS activity updates all LEDs within the system and implement blink functionality as needed.

3.7.2.7 Link Configuration Disable/Resume

The above described flows may be interrupted for debug purposes either by disabling link management for a specific port, disabling link management for all ports via the Set PHY Debug link command or by stopping the Dreadnought Lake scripts at a particular breakpoint. Resuming from such a state is a function of where the breakpoint has been encountered. [Figure 3-28](#) illustrates the cases which result in a reset of the Link Manager flows and the cases which result in resuming the link management flows at the intercept point indicated by "X Return from Brk Pnt w/o Reset" on the Link Configuration Flow diagram ([Figure 3-19](#)).

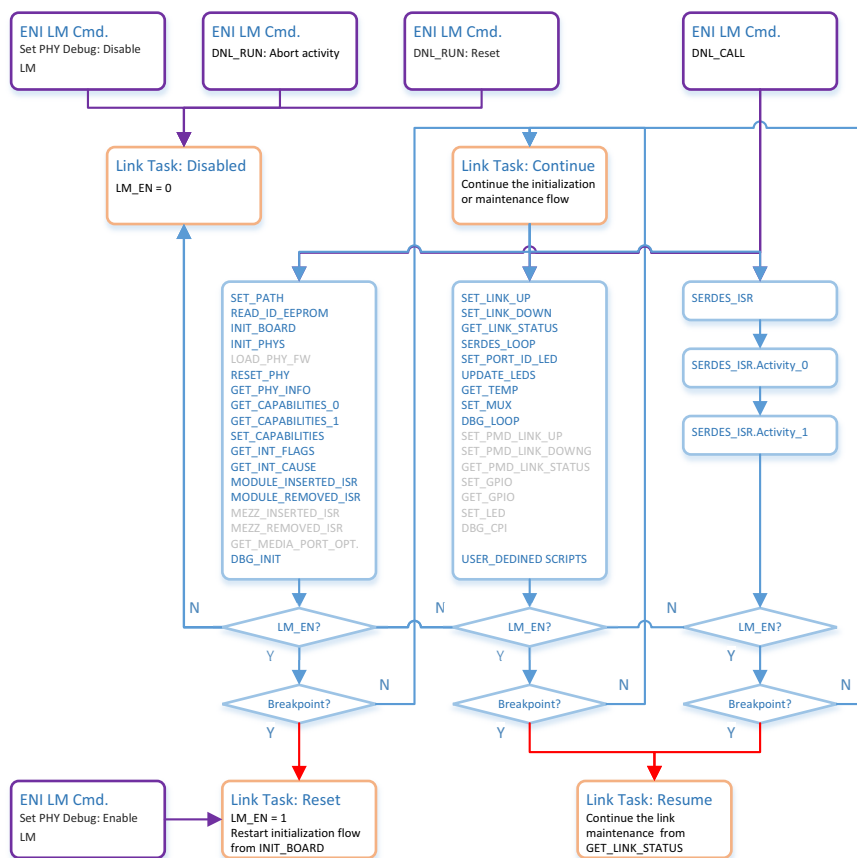


Figure 3-28. Link Configuration Disable/Resume

3.7.3 Link Commands (AQ Commands)

The E610's Link Manager supports the following commands for configuring and managing the link. They provide control over the Ethernet Link Establishment flows, including the Ethernet MAC and Ethernet PHY Subsystems.

The Link Manager provides link configuration and status services to the software driver through these commands. These commands are exposed as AQ Commands.

The software driver can use a link configuration command only for a port that it owns its management. In the E610 the Link Manager allows the software driver control over all ports.

3.7.3.1 Link Commands Summary

Table 3-89 summarizes the link commands. These commands can be used by software for link configuration and maintenance purposes. Typically these commands are used for normal configuration operations.

Table 3-89. Link Commands

Command	Opcode	Description	Section Reference
Get PHY Capabilities	0x0600	Get various PHY capabilities supported by a port.	Section 3.7.3.3
Set PHY Config	0x0601	Sets various PHY configuration parameters of a port.	Section 3.7.3.4
Set MAC Config	0x0603	Sets various MAC configuration parameters of a port.	Section 3.7.3.5
Setup Link and Restart Auto-Negotiation	0x0605	Sets up the link and restarts link auto-negotiation. Note: This command needs to be executed for other set link parameters to take effect on the link.	Section 3.7.3.6
Get Link Status	0x0607	Get the current status of a port.	Section 3.7.3.7
Link Status Event	0x0607	Link status event messages generated by link manager.	Section 3.7.3.8
Set Link Status Event Mask	0x0613	Sets the link status event mask.	Section 3.7.3.9

3.7.3.2 Diagnostics and Maintenance Commands Summary

Table 3-90 summarizes the link diagnostics commands. These can be used by software for diagnostic and maintenance purposes. Typically these commands are not needed for normal run-time operations.

Table 3-90. Diagnostics and Maintenance Commands

Command	Opcode	Description	Section Reference
Set PHY Loopback	0x0619	Sets the PHYs loopback modes of the link.	Section 3.7.3.10
Set PHY Debug	0x0622	Resets the PHY, enables or disables Link Manager.	Section 3.7.3.11
Get PHY Debug	0x0623	Returns the state of the Link Manager.	Section 3.7.3.12

Note: The above listed Set PHY Config, Setup Link and Restart Auto-Negotiation, Set PHY Loopback, and Set PHY Debug commands can bring down a previously-established link.

Note: All reserved fields within the command arguments must be 0.

3.7.3.3 Get PHY Capabilities (0x0600)

This command is used by the software driver to determine the various PHY capabilities supported by a specific port.

3.7.3.3.1 Get PHY Capabilities Command Inputs

Table 3-91 specifies the input arguments of the Get PHY Capabilities command.

Table 3-91. Get PHY Capabilities Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.2.1 for details.
Opcode	2:3.[15:0]	0x0600	Command opcode.
Datalen	4:5.[15:0]		Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:17.[15:0]	0x0	Reserved. Must be 0x0.
Reserved	18:19.[15:4]	0x0	Reserved. Must be 0x0.
Report Mode	18:19.[3:1]		Report Mode 000b = Report topology capabilities, without media Report capabilities as read from the topology netlist. This report ignores the media that might be connected and limits the capabilities. 001b = Report topology capabilities, with media Report capabilities as read from topology netlist, intersected with the detected media capabilities if available. Example: When the topology includes cage but there is no media connected (i.e. the cage is empty), it reports capabilities as read from the topology netlist just like Report Mode 0. When there is specific media connected, i.e. a module is plugged into the cage, it reports only modes/speeds supported by both the topology and the present media. 010b = Report active configuration Report the actively used capabilities configured as it was configured at the last software configuration, MNG configuration or insertion of media and automatic configuration. When the Override enable bit is set at the Link Default Override Mask PFA section, the capabilities are taken from the same PFA section. 011b = Reserved 100b = Report default configuration: This report mode returns the default configuration based on link topology netlist, Link Default Override Mask PFA section, relevant override bits in the PFA section and Strict or Lenient mode. The default configuration does not depend on the software configuration that might be applied using Set PHY Config admin command.
Report Qualified Modules	18:19.[0]		Report Qualified Modules 0b = Do not include the list of qualified modules. 1b = Return the list of qualified modules as part of the response.
Reserved	20:31.[15:0]	0x0	Reserved. Must be 0x0.

3.7.3.3.2 Get PHY Capabilities Command Outputs

Section 3-92 lists output arguments of the Get PHY Capabilities command.

Table 3-92. Get PHY Capabilities Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.2.2 for details.
Opcode	2:3.[15:0]	0x0600	Command opcode.
Datalen	4:5.[15:0]		Set by firmware according to the actual size of the response buffer.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.3.3.3, "Get PHY Capabilities Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:23.[63:0]	0x0	Reserved. Must be 0x0.
Data Address High	24:27.[31:0]		High bits of command buffer address.
Data Address Low	28:31.[31:0]		Low bits of command buffer address.

Table 3-93. Get PHY Capabilities Command Output Buffer

Argument	Byte.[Bits]	Value	Description
PHY Types	0:15.[127:0]		See detailed description in Section 3.8.3.1.1, "Extended PHY Capabilities (128-bit Structure)" .
Tx Pause Ability	16.[0]		Rx Pause Ability The port's default Tx Pause Ability. This parameter may be used by the device driver to determine the IEEE 802.3x PAUSE abilities of the port. 0b = The port does not advertise Tx Pause ability. 1b = The port does advertise Tx Pause ability.
Rx Pause Ability	16.[1]		Rx Pause Ability The port's default Rx Pause Ability. This parameter may be used by the device driver to determine the IEEE 802.3x PAUSE abilities of the port. 0b = The port does not advertise Rx Pause ability. 1b = The port does advertise Rx Pause ability.
Low Power Mode	16.[2]		Low Power Mode 0b = High Power Mode. 1b = Low Power Mode.
Link Mode	16.[3]		Link Mode 0b = Link is disabled. 1b = Link is enabled.
Reserved	16.[4]	0b	Reserved. Must be 0b
Module Qualification Enable	16.[5]		Module Qualification Enable 0b = Pluggable module qualification is disabled. 1b = Pluggable module qualification is enabled.
LESM Enable	16.[6]		LESM Enable 0b = The Link Establishment State Machine is disabled. 1b = The Link Establishment State Machine is enabled.

Table 3-93. Get PHY Capabilities Command Output Buffer [continued]

Argument	Byte.[Bits]	Value	Description
Auto FEC Enable	16.[7]		Automatic FEC Enable 0b = The automatic selection of FEC mode is disabled. 1b = The automatic selection of FEC mode is enabled.
LPLU Enable	17.[0]		LPLU Enable Low Power Link Up (LPLU) Enable for low power (D3Cold) state. 0b = LPLU disabled. Auto-negotiation resolves to the highest common denominator. 1b = LPLU enabled. Auto-negotiation resolves to the lowest common denominator.
AN 28 Enable	17.[1]		AN 28 Enable 0b = AN28 is disabled. 1b = AN28 is enabled. Report based on link mode. See Section 3.8.3.1, “PHY Capabilities” for more detail. Note: Reserved and reads as 0 when Get PHY Capabilities is called with the Report Mode of Report Active Configuration.
AN 73 Enable	17.[2]		AN 73 Enable 0b = AN73 is disabled. 1b = AN73 is enabled. Report based on link mode. See Section 3.8.3.1, “PHY Capabilities” for more detail. Note: Reserved and reads as 0 when Get PHY Capabilities is called with the Report Mode of Report Active Configuration.
AN 37 Enable	17.[3]		AN 37 Enable 0b = AN37 is disabled. 1b = AN37 is enabled. Report based on link mode. See Section 3.8.3.1, “PHY Capabilities” for more detail. Note: Reserved and reads as 0 when Get PHY Capabilities is called with the Report Mode of Report Active Configuration.
Reserved	17.[7:4]	0x0	Reserved. Must be 0x0.
EEE Capability Enable	18:19.[15:0]		EEE Capability Enable Reports the EEE capability for each PHY type supported on the port. One bit per PHY type. These capabilities refers to the outermost PHY connected to E610 link. The following parameter indicates the bit number. Ignores values for unsupported PHY types. This structure is aligned with the “EEE options 1”, in the link topology netlist. Byte.Bit Description 18.0 EEE is enabled for 100BASE-TX 18.1 EEE is enabled for 1000BASE-T 18.2 EEE is enabled for 10GBASE-T 19.3 EEE is enabled for 5GBASE-T 19.4 EEE is enabled for 2.5GBASE-T All other values are reserved and must be zero.
EEER	20:21.[15:0]		EEER Current value of the EEER.TW_SYSTEM register.
PHY ID / SFF Vendor OUI	22:25.[31:0]		PHY ID / SFF Vendor OUI - In E610-MCP, this field returns BASE-T PHY ID (1G or 10G). - In E610-SCP connected to SFP (C2M), this field returns the byte vendor OUI of module. - In E610-SCP (customer control external PHY), this field returns E610 PHY ID. - In E610-SCP (INTEL fully control external PHY), this field returns outermost PHY ID.

Table 3-93. Get PHY Capabilities Command Output Buffer [continued]

Argument	Byte.[Bits]	Value	Description														
PHY FW Version	26:33.[63:0]		PHY Firmware Version This 8-byte field represents the firmware version of the outermost PHY.														
10G KR FEC Ability	34.[0]		10G KR FEC Ability Reflects the 10G KR FEC capability advertisement. 0b = FEC Disabled 1b = FEC Enabled (advertised) For the list of applicable modes, refer to Section 3.8.3.1, “PHY Capabilities” .														
10G KR FEC Request	34.[1]		10G KR FEC Request Reflects the 10G KR FEC capability request. 0b = FEC not requested 1b = FEC requested For the list of applicable modes, refer to Section 3.8.3.1, “PHY Capabilities” .														
No FEC Ability	34.[5]		No FEC Ability Reflects the No FEC capability for AUI modes. 0b = No FEC Disabled 1b = No FEC Enabled For the list of applicable modes, refer to Section 3.8.3.1, “PHY Capabilities” .														
Module Compliance Enforcement	35.[0]		Module Compliance Enforcement Controls lenient or strict link establishment behavior. 0b = Lenient mode When an illegal combination of PHY types are detected, link is attempted using a simplified LESM table including all configured auto-negotiated PHY types and a single AUI PHY type determined based on the configured non negotiated/ optical PHY Types. 1b = Strict mode When an illegal combination of PHY types is detected, link is not configured.														
Reserved	35.[7:1]	0x0	Reserved. Must be 0x0.														
SFF Extended Compliance Code	36.[7:0]		SFF Extended Compliance Code Returns the extended compliance code of the module as defined in SFF specifications (SFP+/SFP28: Addr 0xA0, Offset 36) This field is valid only if <i>Media Available</i> is asserted. It can be used to determine the module type for port when connected to external modules.														
SFF Module Identifier	37.[7:0]		SFF Module Identifier Returns the SFF module identifier as defined by the SFF specifications (SFP+/SFP28: Address 0xA0, Offset 0) This field is valid only if <i>Media Available</i> is asserted.														
SFF Module Type	38.[7:0]		SFF Module Type The following bits might be set to indicate the supported technologies: <table><thead><tr><th>Bit</th><th>Description</th></tr></thead><tbody><tr><td>0</td><td>SFP+ Cu Passive</td></tr><tr><td>1</td><td>SFP+ Cu Active</td></tr><tr><td>4</td><td>10G Base-SR</td></tr><tr><td>5</td><td>10G Base-LR</td></tr><tr><td>6</td><td>10G BASE-LRM</td></tr><tr><td>7</td><td>10G BASE-ER</td></tr></tbody></table> This field is valid only if <i>Media Available</i> is asserted.	Bit	Description	0	SFP+ Cu Passive	1	SFP+ Cu Active	4	10G Base-SR	5	10G Base-LR	6	10G BASE-LRM	7	10G BASE-ER
Bit	Description																
0	SFP+ Cu Passive																
1	SFP+ Cu Active																
4	10G Base-SR																
5	10G Base-LR																
6	10G BASE-LRM																
7	10G BASE-ER																
SFF GbE Compliance Code	39.[7:0]		SFF GbE Compliance Code Returns the GbE Compliance Code as defined by the SFF specification (SFP+/SFP28: Address 0xA0, Offset 6) This field is valid only if <i>Media Available</i> is asserted.														

Table 3-93. Get PHY Capabilities Command Output Buffer [continued]

Argument	Byte.[Bits]	Value	Description												
Qualified Module Count	40.[7:0]		Qualified Module Count Number of qualified modules to be listed in the following bytes. Note: Current definition allows for up to 16 modules												
Cage Type	41.[2:0]		Cage Type 000b = No Cage 001b = SFP+/SFP28 All other values are reserved.												
Reserved	41.[7:3]	0x0	Reserved. Must be 0x0.												
Reserved	42:47.[47:0]	0x0	Reserved. Must be 0x0.												
Qualified Module ID[n]	48:79.[255:0] +32n		Qualified Module ID[n] This is a list of qualified modules that are supported by the E610 and might be connected. Where <i>n</i> represents the index of the qualified module structure and is ranging from 0 to 16. The list contains an entry per module, based on IEEE Std 802.3 definition of device ID, containing: <table><tr><th>Byte</th><th>Description</th></tr><tr><td>0:2</td><td>Vendor OUI</td></tr><tr><td>3</td><td>Reserved</td></tr><tr><td>4:19</td><td>Vendor Part#</td></tr><tr><td>20:23</td><td>Vendor Rev#</td></tr><tr><td>24:31</td><td>Reserved</td></tr></table>	Byte	Description	0:2	Vendor OUI	3	Reserved	4:19	Vendor Part#	20:23	Vendor Rev#	24:31	Reserved
Byte	Description														
0:2	Vendor OUI														
3	Reserved														
4:19	Vendor Part#														
20:23	Vendor Rev#														
24:31	Reserved														

3.7.3.3.3 Get PHY Capabilities Command Errors

The following errors can be returned by the Get PHY Capabilities command:

Table 3-94. Get PHY Capabilities Command Errors

Error Code	Value	Description
OK	0	No error.
EIO	5	Error while accessing the information.
EAGAIN	8	PHY/module interface currently busy, retry.
EINVAL	14	Invalid parameters (for example, buffer too small).

3.7.3.4 Set PHY Config (0x0601)

This command is used to set the various PHY configuration parameters supported on a specific port.

When *Enable Automatic Link Update* is set to 0b, this command must be followed by the Setup Link and Restart Auto-Negotiation command in order for any changes to the link parameters to actually take effect.

When *Enable Automatic Link Update* is set to 1b, the Link Manager returns from the Set PHY Config command after latching in the requested configuration and initiating the link up process, equivalent of the Setup Link and Restart Auto-Negotiation command.

3.7.3.4.1 Set PHY Config Command Inputs

Table 3-95 details the input arguments of the Set PHY Config command.

Table 3-95. Set PHY Config Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.2.1 for details.
Opcode	2:3.[15:0]	0x0601	Command opcode.
Datalen	4:5.[15:0]	0x18	Length of the data buffer in bytes. Must be 0x18
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:23.[63:0]	0x0	Reserved. Must be 0x0.
Data Address High	24:27.[31:0]		High bits of command buffer address.
Data Address Low	28:31.[31:0]		Low bits of command buffer address.

Table 3-96. Set PHY Config Command Input Buffer

Argument	Byte.[Bits]	Value	Description
PHY Types	0:15.[127:0]		PHY Types PHY types (PHY capabilities) supported on port. Describes the capabilities of the outermost PHY, with one bit per PHY Type. PHYs connected to the E610 might be capable of supporting multiple PHY types, indicated using the link topology netlist and the software driver uses this command to indicate the capabilities it wish to enable. This structure is identical to the extended PHY capabilities structure, as described in Section 3.8.3.1.1, "Extended PHY Capabilities (128-bit Structure)". This parameter is used by the software driver to set the various PHY Types to be supported on the port. The port can be configured for a subset of the actual PHY Capabilities available on the port. The actual PHY Capabilities available are read by the device driver using Get PHY Capabilities command. When the Link Establishment State Machine (LESM) is enabled, the E610 iterates over the enabled PHY Types that were provided by the software driver following the LESM rules described in LESM Configuration Rules table until link is established. When the LESM is disabled, this field should enable only a single PHY Type or set of compatible AN PHY Types, which is then used for link establishment. Some of the link parameters, as speed and FEC, may be configured based on the results of auto-negotiation depending on the selected PHY Types

Table 3-96. Set PHY Config Command Input Buffer [continued]

Argument	Byte.[Bits]	Value	Description
Tx Pause En	16.[0]		Tx Pause Enable This parameter is used by the software driver to set the IEEE 802.3x pause ability of the port. The E610's pause ability can be read by using the Get Link Status or Get PHY Capabilities commands. The software driver might disable the IEEE 802.3x link pause ability using this command. If the link is already up and configured, the software driver needs to restart auto-negotiation, so the updated pause ability could be advertised to the link partner in order for the setting to take effect on the link. 0b = Disable Tx Pause ability 1b = Enable Tx Pause ability Note: Simultaneous operation of LFC and PFC is not supported. When PFC is enabled, the PAUSE ability bits are cleared during auto-negotiation regardless of the value of this field. However, if PFC is disabled in the future, the last configured value is used
Rx Pause En	16.[1]		Rx Pause Enable This parameter is used by the software driver to set the IEEE 802.3x pause ability of the port. The E610's pause ability can be read by using the Get Link Status or Get PHY Capabilities commands. The software driver might disable the IEEE 802.3x link pause ability using this command. If the link is already up and configured, the software driver needs to restart auto-negotiation, so the updated pause ability could be advertised to the link partner in order for the setting to take effect on the link. 0b = Disable Rx Pause ability 1b = Enable Rx Pause ability Note: Simultaneous operation of LFC and PFC is not supported. When PFC is enabled, the PAUSE ability bits are cleared during auto-negotiation regardless of the value of this field. However, if PFC is disabled in the future, the last configured value is used.
Low Power Mode	16.[2]		Low Power Mode 0b = High Power Mode 1b = Low Power Mode High Power Mode is not enabled if the port is mapped into a pluggable media with a power class corresponding to a power consumption that exceeds the maximum power allowed for the cage. See <i>Module Max Power Hi</i> and <i>Module Max Power Lo</i> defined within the Node Header Section (Section 3.8.6.2.3).
Enable Link	16.[3]		Enable Link 0b = Disable the link 1b = Enable the link The software driver should not force link down when port is being used for manageability or WoL
Reserved	16.[4]	0b	Reserved. Must be 0b.
Enable Automatic Link Update	16.[5]		Enable Automatic Link Update 0b = Automatic link update disabled The software driver maintains the responsibility for sending the Setup link and restart auto-negotiation command. 1b = Automatic link update enabled The Link Manager automatically executes the Setup Link and Restart Auto-Negotiation command following this command. Note: When automatic link update is enabled, the device driver should be aware that a link change event may occur following the Set PHY Config command.
LESM Enable	16.[6]		LESM Enable 0b = Disable Link Establishment State Machine 1b = Enable Link Establishment State Machine For more detail, see Section 3.7.2.3.1, "Link Establishment State Machine (LESM)".

Table 3-96. Set PHY Config Command Input Buffer [continued]

Argument	Byte.[Bits]	Value	Description
LESM Auto FEC Enable	16.[7]		LESM Automatic FEC Enable 0b = Disable Automatic selection of FEC mode 1b = Enable Automatic selection of FEC mode For more detail, see Section 3.7.2.3.1, "Link Establishment State Machine (LESM)" .
LPLU Enable	17.[0]		LPLU Enable Low Power Link Up (LPLU) Enable for low power (D3Cold) state. 0b = Disable LPLU. Auto-negotiation resolves to the highest common denominator. 1b = Enable LPLU. Auto-negotiation resolves to the lowest common denominator,
Reserved	17.[2:1]	00b	Reserved. Must be 00b.
AN37 Enable	17.[3]		AN37 Enable 0b = Disable Clause 37 auto-negotiation 1b = Enable Clause 37 auto-negotiation Note: The value of the <i>AN37 Enable</i> bit should be propagated from Get PHY Capabilities output to Set PHY Config input for all default software and firmware flows.
Reserved	17.[7:4]	0x0	Reserved. Must be 0x0.
EEE Capability Enable	18:19.[10:0]		EEE Capability Enable Sets EEE capability for each PHY type supported on the port. One bit per PHY type. These capabilities refers to the outermost PHY connected to E610 link. The following parameter indicates the bit number. Ignores values for unsupported PHY types. This structure is aligned with the "EEE options 1", in the link topology netlist. Byte.Bit Description 18.0 EEE is enabled for 100BASE-TX 18.1 EEE is enabled for 1000BASE-T 18.2 EEE is enabled for 10GBASE-T 19.3 EEE is enabled for 5GBASE-T 19.4 EEE is enabled for 2.5GBASE-T All other values are reserved and must be zero. This field is used by the device driver to enable the EEE capability of various PHY types supported on the port. The EEE capability of the E610 can be read by the Get PHY Capabilities command. The device driver might set EEE capability for a subset of PHY types supported by the E610.
Reserved	18:19.[15:11]	0x0	Reserved. Must be 0x0.
EEER	20:21.[15:0]		EEER Value to program the EEER.TW_SYSTEM register. Note: The EEER register is changed immediately with the execution of this command.
10G KR FEC Ability	22.[0]		10G KR FEC Ability Controls the 10G KR FEC capability advertisement. 0b = FEC Disabled 1b = FEC Enabled (advertised) For the list of applicable modes, refer to Section 3.8.3.1, "PHY Capabilities" .
10G KR FEC Request	22.[1]		10G KR FEC Request Controls the 10G KR FEC capability request. 0b = FEC not requested 1b = FEC requested For the list of applicable modes, refer to Section 3.8.3.1, "PHY Capabilities" .

Table 3-96. Set PHY Config Command Input Buffer [continued]

Argument	Byte.[Bits]	Value	Description
No FEC Ability	22.[5]		No FEC Ability Controls the No FEC capability for AUI modes. 0b = No FEC Disabled 1b = No FEC Enabled For the list of applicable modes, refer to Section 3.8.3.1, "PHY Capabilities" .
Module Compliance Enforcement	23.[0]		Module Compliance Enforcement Controls lenient or strict link establishment behavior. 0b = Lenient mode When an illegal combination of PHY types are detected, link is attempted using a simplified LESM table including all configured auto-negotiated PHY types and a single AUI PHY type determined based on the configured non-negotiated/optical PHY Types. 1b = Strict mode When an illegal combination of PHY types are detected link is not configured.
Reserved	23.[7:1]	0x0	Reserved. Must be 0x0.

3.7.3.4.2 Set PHY Config Command Outputs

Table 3-97 specifies the Set Link Set PHY Config command output structure.

Table 3-97. Set PHY Config Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.2.2 for details.
Opcode	2:3.[15:0]	0x0601	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.3.4.3, "Set PHY Config Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:31.[127:0]	0x0	Reserved. Must be 0x0.

3.7.3.4.3 Set PHY Config Command Errors

The following errors can be returned by the Set PHY Config command:

Table 3-98. Set PHY Config Command Errors

Error Code	Value	Description
OK	0	No error.
EPERM	1	Operation is not permitted.
EBUSY	12	Returned when a previous request to program or read an external topology device is ongoing. This code also returned when RDE operation is currently executed.
EMODE	21	Operation not allowed in current device mode. This error is returned when the command is called with disable link, but the link is configured to be used for manageability. In this case, the link is not configured.

3.7.3.5 Set MAC Config (0x0603)

This command is used by to set the various MAC configuration parameters supported on a specific port.

3.7.3.5.1 Set MAC Config Command Inputs

Table 3-99 details the input arguments of the Set MAC Config command.

Table 3-99. Set MAC Config Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.2.1 for details.
Opcode	2:3.[15:0]	0x0603	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes. Must be 0x0
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Max Frame Size	16:17.[15:0]		Max Frame Size This is a 16-bit value used to set the maximum frame size of the Ethernet frame on the port. This parameter is used by the device driver to set the maximum frame size on the port both for Rx and for Tx. This parameter should be set to the maximum expected L2 packet size. It is 1.5 KB or 9.5 KB depending if jumbo packets are expected on the link. The firmware should return ERANGE error when the value provided by the software is bigger than the maximal frame size supported by the MAC (9.5K). Note: The maximal frame size is checked on packets transmitted before the Ethernet CRC is appended to them. For received packets, the check is performed on the packet length that includes the Ethernet CRC.
Reserved	18.[2:0]	000b	Reserved. Must be 000b.
Pacing Config	18.[6:3]		Pacing Config This is 4-bit field that allows configuring PACE parameter in the MAC to slow down the effective data rate. For Fixed IPG pacing, defines the number of 16-bytes words added.
Pacing Type	18.[7]		Pacing Type 0b = Reserved 1b = Fixed IPG pacing
Transmit Timer Priority	19.[7:0]		Transmit Timer Priority This bitmap field selects the priority <n>, with one bit per priority. The priorities selected here are updated with the Transmit Time Value field and the FC refresh threshold field with the values provided in this command. MFP Mode: N/A
Transmit Timer Value	20:21.[15:0]		Transmit Timer Value This is the priority <n> timer value that is included in the XOFF frames being transmitted. <n> is selected in the <i>Transmit Timer Priority</i> bitmap above. <n> = 0 is used for Link Level FC. MFP Mode: See <i>Max Frame Size</i> description above.

Table 3-99. Set MAC Config Command Inputs [continued]

Argument	Byte.[Bits]	Value	Description
FC Refresh Threshold	22:23.[15:0]		FC Refresh Threshold This field represents priority <n> FC refresh threshold, that specifies how many slot time before the XOFF expires, a new XOFF is sent. <n> is selected in the <i>Transmit Timer Priority</i> bitmap above. When <n> = 0, the value is used for link level flow control. This value is used to calculate the actual refresh period for sending the next pause frame if conditions for a pause state are still valid. MFP Mode: PARTIAL 1 = Base line value is loaded from NVM. 2 = Always take MAX value set by all PFs. Note: Firmware should track the baseline value loaded from NVM and the values assigned by PFs. There is no need to remember which PF sets the MAX value. For example, the PF reset does not affect this value.
Auto Drop Blocking Packets	24.[0]		Auto Drop Blocking Packets Controls the behavior when a no-drop packet is blocking a TC queue. 0b = The PF driver is notified. 1b = The blocking packet is dropped and then the PF driver is notified.
Reserved	24.[7:1]	0x0	Reserved. Must be 0x0.
Reserved	25:31.[55:0]	0x0	Reserved. Must be 0x0.

3.7.3.5.2 Set MAC Config Command Outputs

Table 3-100 details the output arguments of the Set MAC Config command.

Table 3-100. Set MAC Config Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.2.2 for details.
Opcode	2:3.[15:0]	0x0603	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.3.5.3, "Set MAC Config Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:31.[127:0]	0x0	Reserved. Must be 0x0.

3.7.3.5.3 Set MAC Config Command Errors

The following errors can be returned by the Set MAC Config command:

Table 3-101. Set MAC Config Command Errors

Error Code	Value	Description
OK	0	No error.
EPERM	1	Operation not permitted.
ERANGE	18	Parameter out of range.

3.7.3.6 Setup Link and Restart Auto-Negotiation (0x0605)

This command is used to setup the link and apply pending Set PHY Config command parameters, restart the auto-negotiation or reset the link in case of non-auto-negotiated links. This command needs to be executed for any change in link parameters, such as set link speed, etc., to take effect.

Note: This command may be executed automatically by the Link Manager following a Set PHY Config command. In such a case these the following arguments are used:

```
Restart Link = 1
Enable Link = Set PHY Config.Enable Link.
```

This command can be used by the software driver to enable or disable the link without changing the other link settings. The link parameters are loaded from the link topology netlist, but the link can be kept down until the software driver issues this command to enable the link with the default settings.

3.7.3.6.1 Setup Link and Restart Auto-Negotiation Command Inputs

Table 3-102 specifies the input arguments of the Setup Link and Restart Auto-Negotiation command.

Table 3-102. Setup Link and Restart Auto-Negotiation Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x0605	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:17.[7:0]	0x0	Reserved. Must be 0x0.
Reserved	18.[0]	0b	Reserved. Must be 0b.
Link Restart	18.[1]		Link Restart 0b = Do not restart the link 1b = Restart the link
Link Enable	18.[2]		Link Enable 0b = Disable link 1b = Enable link
Reserved	18.[7:3]	0x0	Reserved. Must be 0x0.
Reserved	19:31	0x0	Reserved. Must be 0x0.

3.7.3.6.2 Setup Link and Restart Auto-Negotiation Command Outputs

Table 3-103 specifies the output arguments of the Setup Link and Restart Auto-Negotiation command.

Table 3-103. Setup Link and Restart Auto-Negotiation Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x0605	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.3.6.3, "Setup Link and Restart Auto-Negotiation Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:31.[127:0]	0x0	Reserved. Must be 0x0.

3.7.3.6.3 Setup Link and Restart Auto-Negotiation Command Errors

The following errors can be returned by the Setup Link and Restart Auto-Negotiation command:

Table 3-104. Setup Link and Restart Auto-Negotiation Command Errors

Error Code	Value	Description
OK	0	No error.
EBUSY	12	When the command is called when a previous request to program or read the external topology device is ongoing. This code also is returned when RDE operation is executed.
EMODE	21	Operation not allowed in current device mode. This error is returned when the command is called with disable link, but the link is configured to be used for manageability. In this case, the link is not configured.

3.7.3.7 Get Link Status (0x0607)

This command is used to find out the current link status of a port. Link Manager returns link status as Up when the link is available for transmission/reception. This command also returns other operating parameters of the link such as negotiated speed, PHY type, and so on.

3.7.3.7.1 Get Link Status Command Inputs

Table 3-105 specifies the input arguments of the Get Link Status command.

Table 3-105. Get Link Status Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.2.1 for details.
Opcode	2:3.[15:0]	0x0607	Command opcode.
Datalen	4:5.[15:0]		Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:17.[15:0]	0x0	Reserved. Must be 0x0.
Reserved	18.[7:2]	0x0	Reserved. Must be 0x0.
LSE Enable	18.[1:0]		LSE Enable Link Status Event enable configuration 00b = NOP: LSE notification value is not modified and Get link status response returns the most updated value of enable/disable. 01b = Reserved. 10b = Disable link status event notification to software. 11b = Enable link status event notification to software For details on LSE and enabling/disabling LSE events, see Section 3.7.3.8 .
Reserved	19:23.[39:0]	0x0	Reserved. Must be 0x0.
Data Address High	24:27.[31:0]		High bits of command buffer address.
Data Address Low	28:31.[31:0]		Low bits of command buffer address.

3.7.3.7.2 Get Link Status Command Outputs

Table 3-106 lists the output arguments of the Get Link Status command such as link up/down, negotiated/operating speed, fault conditions, and so on.

Table 3-106. Get Link Status Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.2.2 for details.
Opcode	2:3.[15:0]	0x0607	Command opcode.
Datalen	4:5.[15:0]	0x0	Set by firmware according to the actual size of the response buffer.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.3.7.3, "Get Link Status Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.

Table 3-106. Get Link Status Command Outputs [continued]

Argument	Byte.[Bits]	Value	Description
Reserved	16:17.[15:0]	0x0	Reserved. Must be 0x0.
LSE Enable	18.[0]		LSE Enable Link Status Event enable status 0b = Indicates that LSE is disabled. 1b = Indicates that LSE is enabled. For details on LSE and enabling/disabling LSE events, see Section 3.7.3.8 .
Reserved	18.[7:1]	0x0	Reserved. Must be 0x0.
Reserved	19:23	0x0	Reserved. Must be 0x0.
Data Address High	24:27.[31:0]		High bits of command buffer address.
Data Address Low	28:31.[31:0]		Low bits of command buffer address.

Table 3-107. Get Link Status Command Output Buffer

Argument	Byte.[Bits]	Value	Description
Topology and Media Conflicts	0.[7:0]		Topology and Media Conflicts Bit Description 0 Unresolved topology conflict detected. The mezzanine card topology does not match with the main board topology or multiple port options available, but the automatic conflict resolution bit is turned off. 1 Unresolved media conflict detected. The detected media does not match with the available topology options or media supports multiple port options, but the automatic conflict resolution bit is turned off. 2 LOM Topology netlist corrupted. The LOM topology netlist is corrupted and cannot be read from the NVM. 4 Topology netlist load detected Unreachable port. 5 Topology netlist load detected Underutilized port. 6 Topology netlist load detected Underutilized media. 7 Unsupported Media detected. All other values are reserved.
Link Configuration Error	1.[0]		Link Configuration Error 0b = No Link Configuration Error was detected. 1b = Link Configuration Error was detected during attempting to bring link up.
Reserved	1.[1]		Reserved
Invalid Active Port Option Error	1.[2]		Invalid Active Port Option Error Indicates whether the active port option was pointing to a port option marked as invalid in the netlist. 0b = The active port option is valid. 1b = The active port option is invalid.
Invalid Feature ID or Configuration ID Error	1.[3]		Invalid Feature ID or Configuration ID Error Indicates whether the port option in the netlist was pointing to ANVM feature ID or configuration ID that does exist in the current ANVM table. 0b = No invalid Feature ID or Configuration ID were found. 1b = At least one invalid Feature ID or Configuration ID was found.
Critical SDP Error	1.[4]		Critical SDP Error The link topology netlist is missing one or more critical SDP or one or more critical SDP is not configured properly. 0b = No critical SDP errors detected/ 1b = One or more of the TEMP_WARN_N, TEMP_CRIT_N, FAN_ON_AUX indications are either missing from the netlist or not configured properly but marked in NVM as must exist.

Table 3-107. Get Link Status Command Output Buffer [continued]

Argument	Byte.[Bits]	Value	Description										
Module Power Error	1.[5]		Module Power Error Indicates whether the power requirements of the current module exceed the maximum allowed power consumption for the specific cage. 0b = The power consumption of the current module is below the netlist allowed power consumption for the cage. 1b = The power consumption of the current module exceeds the netlist allowed power consumption for the cage. Note: In this case, the module is not enabled and link is not attempted.										
FW Download Error	1.[6]		Firmware Download Error Indicates whether firmware download completed successfully for all relevant PHY nodes within the PHY chain of the port as configured by the link topology netlist. 0b = Firmware download was successfully completed for all relevant PHY nodes. 1b = Firmware download could not successfully be completed for at least one of the PHY nodes with <i>FW Download Enable</i> asserted in the netlist. Note: Possible causes of this issue include missing NVM block holding firmware for nodes with the part number matching the one of the PHY node, non responsive PHY node, and so on.										
Invalid Node Max Power Limit	1.[7]		Invalid Node Max Power Limit 0b = No error detected. 1b = At least one cage node max power limit was overridden by firmware to NVM value or the sum of the max power limit of cage nodes is higher than the board power limit.										
Link Status	2.[0]		Link Status Indicates if the port's link is up and ready for data communication. 0b = Link is down. 1b = Link is up.										
Link Fault	2.[4:1]		Link Fault <table><thead><tr><th>Bit</th><th>Description</th></tr></thead><tbody><tr><td>0</td><td>Returns 1 if PHY has detected a local fault condition, 0 otherwise.</td></tr><tr><td>1</td><td>Returns 1 if a transmit link fault condition is detected, 0 otherwise.</td></tr><tr><td>2</td><td>Returns 1 if a receive link fault condition is detected, 0 otherwise.</td></tr><tr><td>3</td><td>Returns 1 if a remote fault condition detected, 0 otherwise.</td></tr></tbody></table>	Bit	Description	0	Returns 1 if PHY has detected a local fault condition, 0 otherwise.	1	Returns 1 if a transmit link fault condition is detected, 0 otherwise.	2	Returns 1 if a receive link fault condition is detected, 0 otherwise.	3	Returns 1 if a remote fault condition detected, 0 otherwise.
Bit	Description												
0	Returns 1 if PHY has detected a local fault condition, 0 otherwise.												
1	Returns 1 if a transmit link fault condition is detected, 0 otherwise.												
2	Returns 1 if a receive link fault condition is detected, 0 otherwise.												
3	Returns 1 if a remote fault condition detected, 0 otherwise.												
PHY Link Status	2.[5]		PHY Link Status Indicates the status of the PHY link on the outermost link segment. 0b = Link is down. 1b = Link is up.										
Media Available	2.[6]		Media Available 0b = No media is available. 1b = Media is available. This parameter is used by the device driver to find out if the media is available on the port. When connected to an external module, this command returns if the media is plugged in and is available for normal communication. Note: This field is not relevant and returns 1 when the port is not mapped into a pluggable media (i.e. the link topology does not define a cage for the port).										
Signal Detect	2.[7]		Signal Detect 0b = No signal was detected by the PHY or module. 1b = Signal was detected by the PHY or module. In the case of external PHYs, for example, this maps to the global signal detect function in the PHY and in some of the PHYs this maps to energy detect function on the link. In the case of external modules, this maps to the inverse of loss of signal function.										

Table 3-107. Get Link Status Command Output Buffer [continued]

Argument	Byte.[Bits]	Value	Description
AN Completed	3.[0]		AN Completed Returns 1b if auto-negotiation completed successfully or returns 0b otherwise. This bit is valid only if the PHY supports auto-negotiation, auto-negotiation is enabled and the current PHY type configured is requires auto-negotiation.
LP AN Ability	3.[1]		LP AN Ability Returns 1b if the link partner is able to perform auto-negotiation or returns 0b otherwise. This bit is valid only if the PHY supports auto-negotiation and auto-negotiation is enabled.
Parallel Detection Fault	3.[2]		Parallel Detection Fault Returns 1b if the PHY detects parallel detection fault or returns 0b otherwise. This bit is valid only when the PHY supports auto-negotiation with parallel detection enabled or when LESM is enabled. For more information see Section 3.7.2.3.1, "Link Establishment State Machine (LESM)" .
FEC Enabled	3.[3]		FEC Enabled Returns 1b if FEC is enabled on the link or returns 0b otherwise. This bit is valid only for PHY types that support FEC. For PHY types that supports auto-negotiation, FEC might be enabled on the link during auto-negotiation.
Low Power State	3.[4]		Low Power State Returns the Low Power Mode configuration as set by Set PHY Config. 0b = High Power Mode 1b = Low Power Mode
Tx Pause Status	3.[5]		Tx Pause Status 0b = Tx link pause is disabled. 1b = Tx link pause is enabled. This may be the a result of auto-negotiation or a forced configuration for non-negotiated AUI link modes.
Rx Pause Status	3.[6]		Rx Pause Status 0b = Rx link pause is disabled. 1b = Rx link pause is enabled. This may be the a result of auto-negotiation or a forced configuration for non-negotiated AUI link modes.
Qualified Module	3.[7]		Qualified Module When the controller is connected to an external SFP+ module and qualification is required, this field indicates if the module is a qualified module whose OUI matches one of the pre-defined qualified modules. 0b = Module was not found in pre-configured list of qualified modules. 1b = Module is qualified. Note: If module qualification is disabled, this field returns 1.
PHY Temp Alarm	4.[0]		PHY Temp Alarm Returns 1b if a temperature alarm condition is reported by the PHY or returns 0b otherwise. Typically an external PHY generates a temperature alarm condition by signaling a PHY interrupt to firmware. The temperature alarm feature should be enabled in the PHY to generate this condition.
Excessive Link Errors	4.[1]		Excessive Link Errors Returns 1b if an excessive errors over the link condition is reported by the PHY or returns 0b otherwise.
Port Tx Suspended	4.[3:2]		Port Tx Suspended 00b = Port's Tx active. 01b = Port's Tx suspended and drained. 10b = Reserved. 11b = Port's Tx suspended and drained. Blocked TC pipe flushed.
Reserved	4.[7:4]	0x0	Reserved. Must be 0x0.

Table 3-107. Get Link Status Command Output Buffer [continued]

Argument	Byte.[Bits]	Value	Description
Loopback Status	5.[5:0]		Loopback Status This command reports the first PHY in the chain that perform loopback, starting at the outermost. Bit Description 0 PHY local loopback enabled. 1 PHY remote loopback enabled. 2 MAC local loopback enabled. 3:5 The PHY index of the PHY that is reported to perform loopback.
Reserved	5.[7:6]	00b	Reserved. Must be 00b.
Max Frame Size	6:7.[15:0]		Max Frame Size Maximum frame size set on this port.
KR FEC Enabled	8.[0]		KR FEC Enabled Returns 1b if KR FEC was negotiated on the link, 0b otherwise. This field is reported when the current PHY type supports auto-negotiation,
RS FEC 528 Enabled	8.[1]		RS FEC 528 Enabled Returns 1b if RS FEC 528 was negotiated on the link, 0b otherwise. This field is reported when the current PHY type supports auto-negotiation,
Reserved	8.[2]	0b	Reserved. Must be 0b.
Pacing Config	8.[7:3]		Pacing Configuration Bit Description 0:3 This is 4-bit field that enables configuring an average rate pace parameter or fixed IPG pace parameter in the MAC to slow down the effective data rate as listed in TBD. 4 Specifies if the pacing is average rate or fixed IPG. 0b = Reserved 1b = Fixed IPG pacing.
External Device Power Class Ability	9.[1:0]	00b	For SFP: This field contains the supported power ability of the connected module, as follows (bits order [1:0]): 00b = Power Class 1 Module (Low Power) 01b = Power Class 2 Module (High Power) 10b = Power Class 3 Module (High Power) 11b = Power Class 4+ Module (High Power) Note: If QSFP Power Ability is High Power but <Low Power State> is Low Power then link will be disabled For BASE-T (bit 9.0 is as below, bit 9.1 is reserved): 0b = Low and high power. 1b = High power only.
SFF Module Power	9.[7:2]		SFF Module Power Reflects the SFP+/SFP28 module power in 0.5 W increments. The value read from the module is rounded up to the nearest 0.5 W increment. This field is valid only if <i>Media Available</i> is asserted. Note: If the module power class is Power Class 2 or higher and Low Power Mode is asserted, link is disabled.

Table 3-107. Get Link Status Command Output Buffer [continued]

Argument	Byte.[Bits]	Value	Description												
Current Link Speed	10:11.[15:0]		Current Link Speed Returns operating link speed of the port. The PHY might be capable of many speeds but only one speed is enabled as result of configuration or auto-negotiation. This parameter is an 16-bit field, each bit corresponds to a link speed as follows. Only one bit is set at any given time. Valid only when the value of the returned Link Status is Link Is Up. <table><tr><th>Bit</th><th>Description</th></tr><tr><td>1</td><td>100 Mb/s</td></tr><tr><td>2)</td><td>1000 Mb/s</td></tr><tr><td>3</td><td>2.5 Gb/s</td></tr><tr><td>4</td><td>5 Gb/s</td></tr><tr><td>5</td><td>10 Gb/s</td></tr></table> All other values are reserved and must be zero. This parameter is used by the software driver to find out the operating Link speed on the port. The link might be enabled at one of the link speeds due to the result of auto-negotiation/parallel detection or manually configured by Link Manager or software driver when auto-negotiation is disabled. Depending on the device MAC capabilities - some bits might not be set regardless of the PHY because the MAC does not support those speeds.	Bit	Description	1	100 Mb/s	2)	1000 Mb/s	3	2.5 Gb/s	4	5 Gb/s	5	10 Gb/s
Bit	Description														
1	100 Mb/s														
2)	1000 Mb/s														
3	2.5 Gb/s														
4	5 Gb/s														
5	10 Gb/s														
Link Configuration Error Code	12:13.[15:0]		Link Configuration Error Code Reflects the error code returned by the Dreadnought Lake (DNL) scripts. Firmware should report the error of the 'get link status' activity invocation for this admin command if occurred												
Reserved	14.[7:0]	0x0	Reserved. Must be 0x0.												
Link Attempt Counter	15.[7:0]		Link Attempt Counter The Link Attempt Counter reflects the number of iterations the Link Establishment State Machine has made since the last Setup Link and Restart Auto-Negotiation command. The counter is cleared on every Setup Link and Restart Auto-Negotiation command and does not roll over.												
Current PHY Type	16:31.[127:0]		Current PHY Type This field has a single bit set to indicate the PHY type per the link's mode of operation. The firmware should report back only the current PHY types and speed, based on topology option or the latest result of the auto-negotiation. The byte encoding is identical to PHY type (PHY capabilities) field found in Section 3.8.3.1.1, "Extended PHY Capabilities (128-bit Structure)" .)												
Reserved	32:55	0x0	Reserved. Must be 0x0.												

3.7.3.7.3 Get Link Status Command Errors

The following errors can be returned by the Get Link Status command:

Table 3-108. Get Link Status Command Errors

Error Code	Value	Description
OK	0	No error.
EIO	5	Error while accessing the information
EAGAIN	8	PHY/module interface currently busy, retry
EINVAL	14	EINVAL. Invalid parameters, e.g., buffer too small. In this case, the link is not configured.

3.7.3.8 Link Status Event (0x0607)

Link Status Events (LSE) are generated by Link Manager whenever there is a change in status in any of the defined below conditions. The software driver can subscribe to or unsubscribe from any or all of these events as described below. When link status events are enabled, any un-masked event triggers the link manager to post a link status event message to the AQ command receive queue. The structure of these messages is identical to the output of the Get Link Status command.

LSE are disabled by default. They need to be explicitly enabled by the software driver setting the LSE Enable input of the Get Link Status command. The Link Manager disables the LSE immediately after generating an LSE and does not queue further events until LSE is explicitly enabled by the software driver via the Get Link Status command. The LSE enabled status is indicated through the LSE Enable bit in the Get Link Status output arguments.

The software driver is expected to subscribe to the relevant LSE generated by Link Manager and to use any hardware link status interrupt mechanisms. Hardware link status change interrupts are provided only for diagnostic use. Hence, hardware link status interrupts should be disabled for normal operation.

Table 3-109. Link Status Events

Event	Mask Bit	Description
Link Change	1	Link state changed. For example, the link state changes from link up to link down.
Media Not Available	2	Event is reported when an external module is pulled out of its cage.
Link Fault Change	3	Link fault conditions changed.
PHY Temperature Alarm Detected	4	Event is generated when an external PHY or module generates a temperature alarm interrupt.
Excessive Errors Detected	5	Excessive Error (High BER) condition was detected.
Signal Detect Change	6	Signal detect state changed.
Auto-Negotiation Completed	7	Auto-negotiation completed.
Module Qualification Failure Detected	8	When working with external modules, firmware might be enabled to perform a validation process where the module ID parameters are compared with a per-configured, NVM loaded, list of qualified modules. If, qualification check is enabled and connected module is not found in the list, then firmware terminates.
Port Tx Suspended	9	Indicates that the port's Tx data path is temporarily suspended for configuration purposes.
Topology Conflict Detected	10	Indicates a conflict in the topology netlist. For more detail, see Section 3.7.6.4.1, "Detecting Conflicts" .
Media Conflict Detected	11	Indicates a conflict caused by the present media. For more detail, see Section 3.7.6.4.1, "Detecting Conflicts" .

3.7.3.8.1 Link Status Event Output

The structure of the link status messages is identical to the output of the Get Link Status command. See [Section 3.7.3.7.2](#) for more detail.

3.7.3.8.2 Link Status Event Errors

The structure of the link status messages is identical to the output of the Get Link Status command. See [Section 3.7.3.7.3](#) for more detail.

3.7.3.9 Set Link Status Event Mask (0x0613)

This command allows the software driver to mask the various events from potentially causing the link manager to send Link Status Event messages.

3.7.3.9.1 Set Link Status Event Mask Command Inputs

Table 3-110 specifies the Set Link Status Event Mask command input structure.

Table 3-110. Set Link Status Event Mask Command Inputs

Argument	Byte.[Bits]	Value	Description																										
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.																										
Opcode	2:3.[15:0]	0x0613	Command opcode.																										
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.																										
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.																										
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.																										
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.																										
Reserved	16:23.[127:0]	0x0	Reserved. Must be 0x0.																										
Event Mask	24:25.[15:0]		Link Status Event Masks The bit mask might be used to mask one or more event causing conditions. Set bit(s) to 1b to mask an event from causing LSE or set to 0b otherwise. The bits are cleared by default. <table><thead><tr><th>Bit</th><th>Description</th></tr></thead><tbody><tr><td>1</td><td>Mask link up/down condition</td></tr><tr><td>2</td><td>Mask media not available or module not present condition</td></tr><tr><td>3</td><td>Mask link fault condition</td></tr><tr><td>4</td><td>Mask PHY temperature alarm condition</td></tr><tr><td>5</td><td>Mask excessive errors over the link condition</td></tr><tr><td>6</td><td>Mask signal detect (asserted or de-asserted) condition</td></tr><tr><td>8</td><td>Mask module qualification failure condition</td></tr><tr><td>7</td><td>Mask auto-negotiation completed condition</td></tr><tr><td>9</td><td>Mask port Tx suspended condition</td></tr><tr><td>10</td><td>Mask topology conflict condition</td></tr><tr><td>11</td><td>Mask media conflict condition</td></tr><tr><td>12</td><td>PHY firmware load failed</td></tr></tbody></table> All other values are reserved.	Bit	Description	1	Mask link up/down condition	2	Mask media not available or module not present condition	3	Mask link fault condition	4	Mask PHY temperature alarm condition	5	Mask excessive errors over the link condition	6	Mask signal detect (asserted or de-asserted) condition	8	Mask module qualification failure condition	7	Mask auto-negotiation completed condition	9	Mask port Tx suspended condition	10	Mask topology conflict condition	11	Mask media conflict condition	12	PHY firmware load failed
Bit	Description																												
1	Mask link up/down condition																												
2	Mask media not available or module not present condition																												
3	Mask link fault condition																												
4	Mask PHY temperature alarm condition																												
5	Mask excessive errors over the link condition																												
6	Mask signal detect (asserted or de-asserted) condition																												
8	Mask module qualification failure condition																												
7	Mask auto-negotiation completed condition																												
9	Mask port Tx suspended condition																												
10	Mask topology conflict condition																												
11	Mask media conflict condition																												
12	PHY firmware load failed																												
Reserved	26:31.[47:0]	0x0	Reserved. Must be 0x0.																										

3.7.3.9.2 Set Link Status Event Mask Command Outputs

Table 3-111 specifies the Set Link Status Event Mask command output structure.

Table 3-111. Set Link Status Event Mask Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x0613	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes. Must be 0x0.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.3.9.3, "Set Link Status Event Mask Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:31.[127:0]	0x0	Reserved. Must be 0x0.

3.7.3.9.3 Set Link Status Event Mask Command Errors

The following errors can be returned by the Set Link Status Event Mask command:

Table 3-112. Set Link Status Event Mask Command Errors

Error Code	Value	Description
OK	0	No error.

3.7.3.10 Set PHY Loopback (0x0619)

This command is used to set the different loopback modes on the one of the PHYs connected to the port. This command is used for diagnostic or monitoring purposes only. The command should not be executed during normal operation as this may disrupt link operation.

Note: When setting a loopback mode, a single PHY Type must be selected. Only one loopback of any type or level is allowed on a particular port at any given time. Enabling both MAC and PHY loopbacks might result in unexpected behavior.

3.7.3.10.1 Set PHY Loopback Command Inputs

Table 3-113 specifies the Set PHY Loopback command input structure.

Table 3-113. Set PHY Loopback Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x0619	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Logical Port Number	16-17		Logical Port Number 17.0 - Port number is valid 17.1-17.7 - Reserved 16 - Logical Port number. This field specifies the logical port number, and it is used when the physical function owns more than one port. Note: In LKV mode, when bit 17.0 is 0, firmware will use the port owned by the physical function. When bit 17.0 is 1, firmware will use the port indicated in this field.
PHY Index	18.[7:0]		PHY Index The requested PHY index. PHYs are indexed according to their place in the PHY chain, while 0 means the outermost PHY.
Loopback Enable	19.[0]		Loopback Enable Turn on/off the loopback. 0b = Disable loopback 1b = Enable the specified loopback
Loopback Type	19.[1]		Loopback Type Selects the loopback type. 0b = Local (host-side) loopback. Packets transmitted from the host are looped back to the host at the PHY. 1b = Remote (line-side) loopback. Packets received at the line-side PHY's interface are looped back to the link partner.
Loopback Level	19.[2]		Loopback Level Selects the loopback level. 0b = PMD, the loopback is done at the PMD level 1b = PCS, the loopback is done at the PCS level Note: The exact place of the loopback might be PHY dependent/
Reserved	19.[7:3]	0x0	Reserved. Must be 0x0.
Reserved	20:31.[95:0]	0x0	Reserved. Must be 0x0.

3.7.3.10.2 Set PHY Loopback Command Outputs

Table 3-114 specifies the Set PHY Loopback command output structure.

Table 3-114. Set PHY Loopback Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x0619	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes. Must be 0x0.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.3.10.3, "Set PHY Loopback Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:31.[127:0]	0x0	Reserved. Must be 0x0.

3.7.3.10.3 Set PHY Loopback Command Errors

The following errors can be returned by the Set PHY Loopback command:

Table 3-115. Set PHY Loopback Command Errors

Error Code	Value	Description
OK	0	No error.
EPERM	1	Operation not permitted.

3.7.3.11 Set PHY Debug (0x0622)

This command is used to reset a specific PHY associated with a given port and to enable/disable the Link Manager.

3.7.3.11.1 Set PHY Debug Command Inputs

Table 3-116 specifies the input arguments of the Set PHY Debug command.

Table 3-116. Set PHY Debug Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x0622	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Logical Port Number	16-17		Logical Port Number 17.0 - Logical Port number is valid 17.1-17.7 - Reserved 16 - Logical Port number. This field specifies the logical port number, and it is used when the physical function owns more than one port.
PHY Index	18.[7:0]		PHY Index The index of the PHY within the PHY chain of the specified port the PHY Reset should apply to. PHYs are indexed according to their place in the PHY chain with index 0 corresponding to the outermost PHY/
Reserved	19.[0]	0b	Reserved. Must be 0b.
PHY Reset	19.[2:1]		PHY Reset Type 00b = NOP 01b = Hard Reset (Setting the signal that is connected to the reset port of the PHY). 10b = Soft Reset (Setting the PHYs reset register using the register access protocol for this PHY) (for example, I ² C or MDIO for external PHYs). All other values are reserved. Note: Some PHYs may only use one type of reset. (for example, the internal PHY does not have hard reset). Note: PHY Reset requires Logical Port Number to be valid. Returns an error otherwise
Link Manager Port Disable	19.[3]		Link Manager Port Disable Controls Link Management for the specified logical port. The configuration is applied only when the logical port number is valid. 0b = Enable Link Management for the specified port. 1b = Disable Link Management for the specified port. Module hot-plug event triggered flows are not disabled. Inserted modules are initialized and the related port capabilities discovered. For more detail see Section 3.7.2.7, "Link Configuration Disable/Resume" .

Table 3-116. Set PHY Debug Command Inputs [continued]

Argument	Byte.[Bits]	Value	Description
Link Manager All Disable	19.[4]		Link Manager All Disable Controls Link Management for all ports. The configuration is applied only when the logical port number is invalid. 0b = Enable Link Management for the all ports 1b = Disable Link Management for the all ports. All Link Management flows are disabled including flows triggered by module hot-plug events. Note: When link management is disabled for ALL links, all link commands, besides "Get Link Status" are disabled and EINVAL is returned until link management is enabled. For more detail see Section 3.7.2.7, "Link Configuration Disable/Resume" .
Load PHY Firmware	19.[5]		Load PHY Firmware Initiates PHY firmware download from NVM. 0b = No action. 1b = Initiate PHY firmware download. If the PHY node has its <i>Load PHY FW</i> bit asserted in the netlist and there is a section in the NVM with a loadable PHY image that matches the PHY P/N, the Link Manager loads the matching image from NVM to the PHY Note: Loading PHY firmware is only supported in NIC mode. When the system is in switch mode, the request in netlist to load PHY firmware is silently ignored.
Keep PHY Management Enabled	19.[6]		Keep PHY Management Enabled When this bit is set the PHY management scripts (SERDES ISR and SERDES LOOP) are still called for all ports.
Reserved	19.[7]	0b	Reserved. Must be 0b.
Reserved	20:31.[95:0]	0x0	Reserved. Must be 0x0

3.7.3.11.2 Set PHY Debug Command Outputs

Table 3-117 specifies the Set PHY Debug command output structure.

Table 3-117. Set PHY Debug Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x0622	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes. Must be 0x0.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.3.11.3, "Set PHY Debug Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:31.[127:0]	0x0	Reserved. Must be 0x0.

3.7.3.11.3 Set PHY Debug Command Errors

The following errors can be returned by the Set PHY Debug command:

Table 3-118. Set PHY Debug Command Errors

Error Code	Value	Description
OK	0	No error.
ENXIO	6	No such resource. Firmware did not find a valid external topology device image in the NVM to match the requested device.
EBUSY	12	Returned when a previous request to program or read an external topology device is ongoing. This code also returned when RDE operation is currently executed.
ENOSYS	17	Device firmware loading not implemented for this device or the device image in NVM is not enabled for loading. This error is also returned when firmware fails to access the PHY (for example, bus error).

3.7.3.12 Get PHY Debug (0x0623)

This command is used to retrieve the state of the Link Manager.

3.7.3.12.1 Get PHY Debug Command Inputs

Table 3-119 specifies the input arguments of the Get PHY Debug command.

Table 3-119. Get PHY Debug Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x0623	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:31.[127:0]	0x0	Reserved. Must be 0x0.

3.7.3.12.2 Get PHY Debug Command Outputs

Table 3-120 specifies the output arguments of the Get PHY Debug command.

Table 3-120. Get PHY Debug Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x0623	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes. Must be 0x0.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.3.12.3, "Get PHY Debug Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:18.[23:0]	0x0	Reserved. Must be 0x0.
Reserved	19.[7:5]	000b	Reserved. Must be 000b.
Link Manager All Disable	19.[4]		Link Manager All Disable 0b = The Link Management is enabled for the all ports. 1b = The Link Management is disabled for the all ports. Note: When link management is disabled for ALL links, all link commands, besides "Get Link Status" are disabled and EINVAL is returned until link management is enabled.
Link Manager Port Disable	19.[3]		Link Manager Port Disable 0b = The Link Management is enabled for the specified port. 1b = The Link Management is disabled for the specified port.
Reserved	19.[2:0]	000b	Reserved. Must be 000b.
Reserved	20:31.[95:0]	0x0	Reserved. Must be 0x0.

3.7.3.12.3 Get PHY Debug Command Errors

The following errors can be returned by the Get PHY Debug command:

Table 3-121. Get PHY Debug Command Errors

Error Code	Value	Description
OK	0	No error.

3.7.4 SyncE Flows

3.7.4.1 Overview

This section defines the system level SyncE clock management related flows, the assumptions they make, their dependencies and the different firmware and software entities involved. These flows include the configuration and monitoring of PHY recovered clock outputs, DPLLs, and their inputs and outputs.

The task of SyncE clock management is split between firmware and software. That is, the Link Manager and the `sync4l` application.

- The Link Manager provides an abstraction layer for the configuration and monitoring of PHY recovered clock outputs, DPLLs, and their inputs and outputs. It is responsible for initialization of the SyncE related inputs, outputs, and their configuration on the relevant link events. The managed PHY and DPLL devices, their management interfaces and clock connections as well as related configuration information are captured within the link topology netlist as Clock Controller nodes.
- The `sync4l` application is responsible for processing the Ethernet Synchronization Messaging Channel (ESMC) traffic. It uses the received ESMC packets to trace the source of recovered clocks, makes decisions on which ports to source recovered clock outputs from, assigns priorities for the DPLL inputs, and generates ESMC packets to advertise the resulting clock quality on the active ports.

The interface between the Link Manager and the `sync4l` application is defined in [Section 3.7.5](#).

3.7.4.2 Resets

When power is applied it is expected that all on board DPLL devices come out of reset with a default configuration loaded from their incorporated Flash. This configuration includes input and output buffer types, tolerance settings, bandwidth configuration, hysteresis settings, synthesizer frequencies, and so on. These devices are not reset by the init board activity and only the small subset of configuration is updated by the defined Link Manager flows.

When coming out of reset it is expected that all PHY devices have their recovered clock outputs disabled.

3.7.4.3 Initialization

During initialization all clock I/O connections need to be configured according to their corresponding Mode configuration from the link topology netlist. The Link Manager applies the initial configuration in the following order: first the PHY recovered clock outputs, second the DPLL clock reference inputs and last the DPLL clock outputs.

[Figure 3-29](#) illustrates the initialization loops implemented within the Link Manager.

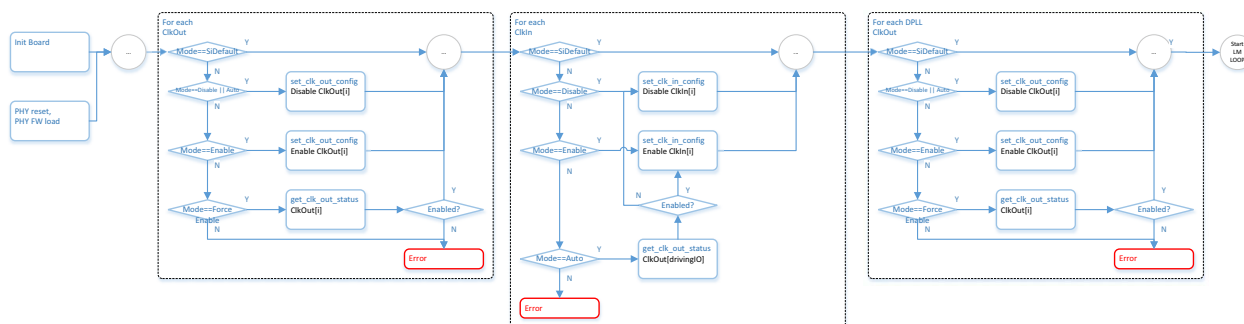


Figure 3-29. SyncE Initialization

Note: This initial configuration is carried out within the init board activity. However, the configuration of PHY recovered clock outputs is also triggered after every reset of the PHY (either part of the init flow, PHY firmware load, or reset initiated via the Set PHY Debug command).

3.7.4.4 Configuration Flow

Once initialization is complete, the Link Manager attempts link on the relevant ports as configured by the link topology netlist. For each link that has been established, **synce4l** determines the quality of the clock they are traceable to and make a decision on which of the ports to source recovered clocks from and configures it via the SyncE commands. This information is then used by the Link Manager to configure the muxes and dividers in the path of the recovered clocks as well as the connected DPLL inputs. Based on the quality of the clocks the ports are traceable to synce4l also makes decisions on the priority of the DPLL inputs, and configures the DPLL to automatically switch between these sources (these may include recovered clocks as well as external or on-board GPS clocks) or instructs it to enter holdover or freerun on the high precision local oscillator. The application monitors the status of the ports as well as the DPLL and this configuration is updated on every change in link status or clock quality detected. However, the change from one clock source to another is intended to happen automatically as the DPLL qualifies or disqualifies the configured inputs.

3.7.4.4.1 Link Status Changes

When link status changes the configuration of associated reference clock outputs and the configuration of the reference clock inputs driven by them needs to be automatically updated. This update flow is triggered by a link status change detected based on get link status output regardless of the condition resulting in the link status change.

When a link up event is detected, if there is a recovered clock output associated with the port then the output is enabled and its divider is updated with the divider value stored in the netlist corresponding to the current baud rate. If there is any IO connections captured in the netlist for this clock output then the connected clock input is enabled and its expected frequency is set based on the current baud rate and configured output divider value.

When a link down event is detected, any associated recovered clock outputs and potentially connected clock inputs are disabled.

Figure 3-30 illustrates the configuration applied in the event of link status changes.

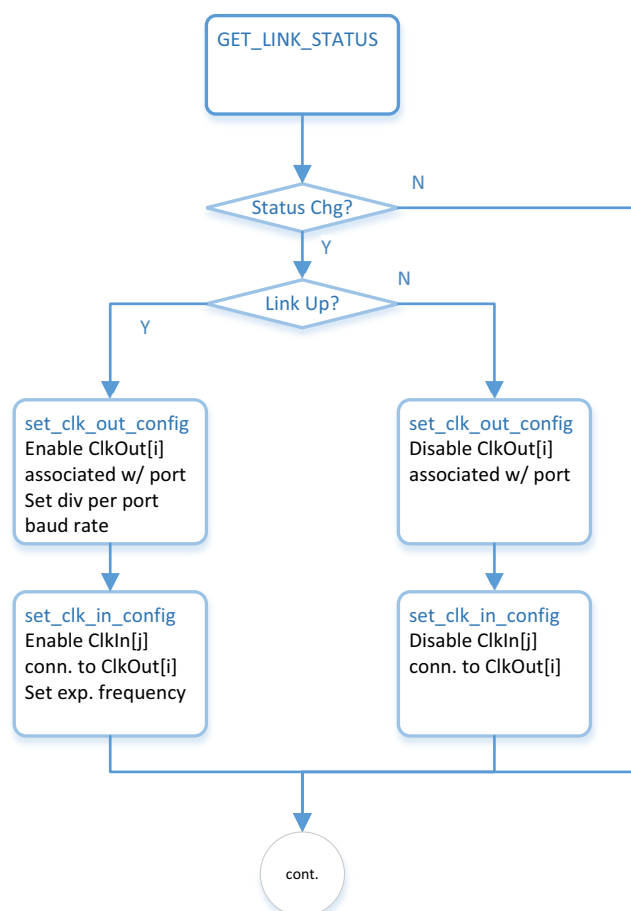


Figure 3-30. SyncE Link Status Changes

3.7.5 SyncE Commands

The E610's Link Manager supports the following commands for configuring and managing the SyncE clocks associated with the PHY and Clock Controller (DPLL/Synthesized/Divider) nodes within the link topology netlist. They provide control and status of available clock inputs, recovered clocks, associated cleanup PLLs and clock outputs.

The Link Manager provides SyncE clock configuration and status services to the software driver through these commands exposed as AQ Commands.

3.7.5.1 SyncE Commands Summary

Table 3-122 summarizes the clock configuration commands. These commands can be used by software for SyncE clock configuration and maintenance purposes. Typically these commands are used for normal configuration operations.

Table 3-122. SyncE Commands

Command	Opcode	Description	Section Reference
Set PHY Recovered Clock Configuration	0x0630	Applies new configuration for the specified recovered clock output.	Section 3.7.5.2
Get PHY Recovered Clock Configuration	0x0631	Returns the current configuration of the specified recovered clock output.	Section 3.7.5.3
Get Clock Controller Capabilities	0x0C61	Retrieves the clock configuration capabilities.	Section 3.7.5.4
Set Clock Controller Input Configuration	0x0C62	Applies new configuration for the specified clock input.	Section 3.7.5.5
Get Clock Controller Input Configuration	0x0C63	Returns the current configuration of the specified clock input.	Section 3.7.5.6
Set Clock Controller Output Configuration	0x0C64	Applies new configuration for the specified clock output.	Section 3.7.5.7
Get Clock Controller Output Configuration	0x0C65	Returns the current configuration of the specified clock output.	Section 3.7.5.8
Get Clock Controller DPLL Status	0x0C66	Returns the current status and configuration of the specified DPLL.	Section 3.7.5.9
Set Clock Controller DPLL Configuration	0x0C67	Applies new configuration for the specified DPLL.	Section 3.7.5.10
Set Clock Controller Reference Priority	0x0C68	Applies new priority configuration for the specified clock input.	Section 3.7.5.11
Get Clock Controller Reference Priority	0x0C69	Returns the current priority configuration of the specified clock output.	Section 3.7.5.12
Get Clock Controller Info	0x0C6A	Retrieves stepping and firmware versions from the clock controller node.	Section 3.7.5.13

3.7.5.2 Set PHY Recovered Clock Configuration (0x0630)

This command maps any of the two Ethernet lanes (PHY Port number) onto the two recovered clocks (PHY output).

3.7.5.2.1 Set PHY Recovered Clock Configuration Command Inputs

Table 3-123 details the input arguments of the Set PHY Recovered Clock Configuration command.

Table 3-123. Set PHY Recovered Clock Configuration Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x0630	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
PHY Output Index	16.[7:0]		PHY Output Index
PHY Port Number (Lane Number)	17.[7:0]		PHY Port Number The PHY lane number associate with the specified output. Note: A value of 0xFF causes the first lane of the port associate with the PF to be used.
Reserved	18.[7:1]	0x0	Reserved. Must be 0x0.
Output Enable	18.[0]		Output Enable 0b = Disable the recovered clock output. 1b = Enable the recovered clock output.
Reserved	19:29.[87:0]	0x0	Reserved. Must be 0x0.
Reserved	30:31.[15:11]	0x0	Reserved. Must be 0x0.
Node Handle Valid	30:31.[10]		Node Handle Valid 0b = The configuration is applied to the outermost PHY node (i.e. PHY[0]) in the PHY chain of the port associated with the PF the command is issued from. Note: The netlist needs to capture two nodes for the outermost PHY device: one with Node Type of PHY and one with Node Type of Clock Controller and a valid Node Parent section pointing to the corresponding former PHY node. The command shall use the <i>Node Handle</i> of the latter, the node with Node Type of Clock Controller 1b = The configuration is applied to the PHY node identified by the <i>Node Handle</i> input.
Node Handle	30:31.[9:0]		Node Handle The handle of the PHY node.

3.7.5.2.2 Set PHY Recovered Clock Configuration Command Outputs

Table 3-124 details the output arguments of the Set PHY Recovered Clock Configuration command.

Table 3-124. Set PHY Recovered Clock Configuration Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x0630	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes. Must be 0x0.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.5.2.3, "Set PHY Recovered Clock Configuration Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
PHY Output Index	16.[7:0]		PHY Output Index
PHY Port Number (Lane Number)	17.[7:0]		PHY Port Number The number of the PHY port associate with the specified output.
Reserved	18.[7:1]	0x0	Reserved. Must be 0x0.
Output Enable	18.[0]		Output Enable 0b = The recovered clock output is disabled. 1b = The recovered clock output is enabled.
Reserved	19:31.[103:0]	0x0	Reserved. Must be 0x0.

3.7.5.2.3 Set PHY Recovered Clock Configuration Command Errors

The following errors can be returned by the Set PHY Recovered Clock Configuration command:

Table 3-125. Set PHY Recovered Clock Configuration Command Errors

Error Code	Value	Description
OK	0	No error.
ENOENT	2	No such element. - Node handle was not found (for example, Index was higher than the number of handles found for the given context). - Uninitialized chain of PHYs for the given port.
EBUSY	12	PHY/module interface currently busy, retry.
EINVAL	14	Invalid parameters (for example, buffer too small).
NOTINIT	30	LM not initialized.
EFATAL_Initialization	31	LM initialization failure.
EFAIL	255	General error.

3.7.5.3 Get PHY Recovered Clock Configuration (0x0631)

This command reads the mapping of the Ethernet lanes to the recovered clocks.

3.7.5.3.1 Get PHY Recovered Clock Configuration Command Inputs

Table 3-126 details the input arguments of the Get PHY Recovered Clock Configuration command.

Table 3-126. Get PHY Recovered Clock Configuration Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x0631	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
PHY Output Index	16.[7:0]		PHY Output Index
Reserved	17:29.[103:0]	0x0	Reserved. Must be 0x0.
Reserved	30:31.[15:11]	0x0	Reserved. Must be 0x0.
Node Handle Valid	30:31.[10]		Node Handle Valid 0b = The configuration is retrieved from the outermost PHY node (i.e. PHY[0]) in the PHY chain of the port associated with the PF the command is issued from. Note: The netlist needs to capture two nodes for the outermost PHY device: one with Node Type of PHY and one with Node Type of Clock Controller and a valid Node Parent section pointing to the corresponding former PHY node. The command shall use the <i>Node Handle</i> of the latter, the node with Node Type of Clock Controller 1b = The configuration is retrieved from the PHY node identified by the <i>Node Handle</i> input.
Node Handle	30:31.[9:0]		Node Handle The handle of the PHY node.

3.7.5.3.2 Get PHY Recovered Clock Configuration Command Outputs

Table 3-127 details the output arguments of the Get PHY Recovered Clock Configuration command.

Table 3-127. Get PHY Recovered Clock Configuration Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x0631	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes. Must be 0x0.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.5.3.3, "Get PHY Recovered Clock Configuration Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
PHY Output Index	16.[7:0]		PHY Output Index

Table 3-127. Get PHY Recovered Clock Configuration Command Outputs [continued]

Argument	Byte.[Bits]	Value	Description
PHY Port Number (Lane Number)	17.[7:0]		Port Number The number of the PHY port associated with the specified output.
Reserved	18.[7:1]	0x0	Reserved. Must be 0x0.
Output Enable	18.[0]		Output Enable 0b = The recovered clock output is disabled. 1b = The recovered clock output is enabled.
Reserved	19:29.[87:0]	0x0	Reserved. Must be 0x0.
Driven Clk Num	30:31.[15:10]		Driven Clock Number The index of the clock input on the node identified by <i>Driven Node Handle</i> the clock output is connected to.
Driven Node Handle	30:31.[9:0]		Driven Node Handle The handle of the node the specified clock output is connected to.

3.7.5.3.3 Get PHY Recovered Clock Configuration Command Errors

The following errors can be returned by the Get PHY Recovered Clock Configuration command:

Table 3-128. Get PHY Recovered Clock Configuration Command Errors

Error Code	Value	Description
OK	0	No error.
ENOENT	2	No such element. - Node handle was not found (for example, Index was higher than the number of handles found for the given context). - Uninitialized chain of PHYs for the given port.
EBUSY	12	PHY/module interface currently busy, retry.
EINVAL	14	Invalid parameters (for example, buffer too small).
NOTINIT	30	LM not initialized.
EFATAL_Initialization	31	LM initialization failure.
EFAIL	255	General error.

3.7.5.4 Get Clock Controller Capabilities (0x0C61)

This command reads the capabilities of the Clock Controller. If the value is not defined or cannot be evaluated, it is 0xFF for 8-bit fields and 0xFFFF FFFF for 32-bit fields.

3.7.5.4.1 Get Clock Controller Capabilities Command Inputs

Table 3-129 details the input arguments of the Get Clock Controller Capabilities command.

Table 3-129. Get Clock Controller Capabilities Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.2.1 for details.
Opcode	2:3.[15:0]	0x0C61	Command opcode.
Datalen	4:5.[15:0]		Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:21.[47:0]	0x0	Reserved. Must be 0x0.
Reserved	22:23[15:11]	0x0	Reserved. Must be 0x0.
Node Handle Valid	22:23.[10]		Node Handle Valid 0b = The configuration is retrieved from the first Clock Controller node in the netlist (i.e. Clock Controller with index 0). 1b = The configuration is retrieved from the Clock Controller node identified by the <i>Node Handle</i> input.
Node Handle	22:23.[9:0]		Node Handle The handle of the Clock Controller node.
Data Address High	24:27.[31:0]		High bits of command buffer address.
Data Address Low	28:31.[31:0]		Low bits of command buffer address.

3.7.5.4.2 Get Clock Controller Capabilities Command Outputs

Table 3-130 details the output arguments of the Get Clock Controller Capabilities command.

Table 3-130. Get Clock Controller Capabilities Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.2.2 for details.
Opcode	2:3.[15:0]	0x0C61	Command opcode.
Datalen	4:5.[15:0]		Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.5.4.3, "Get Clock Controller Capabilities Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:23.[63:0]	0x0	Reserved. Must be 0x0.
Data Address High	24:27.[31:0]		High bits of command buffer address.
Data Address Low	28:31.[31:0]		Low bits of command buffer address.

Table 3-131. Get Clock Controller Capabilities Command Output Buffer

Argument	Byte.[Bits]	Value	Description
Number of Clock Controller Inputs	0.[7:0]		Number of Clock Controller Inputs
Number of Clock Controller Outputs	1.[7:0]		Number of Clock Controller Outputs
PPS DPLL Index	2.[7:0]		PPS DPLL Index Index of the PPS DPLL.
Sync EDPLL Index	3.[7:0]		Sync EDPLL Index Index of the SyncE DPLL.
Maximum Input Frequency	4:7.[31:0]		Maximum Input Frequency Maximum Input Frequency supported.
Maximum Input Phase Adjustment	8:11.[31:0]		Maximum Input Phase Adjustment Maximum input phase adjustment in ps.
Maximum Output Frequency	12:15.[31:0]		Maximum Output Frequency Maximum output frequency.
Maximum Output Phase Adjustment	16:19.[31:0]		Maximum Output Phase Adjustment Maximum output phase adjustment in ps.
Node Part Number	20.[7:0]		Node Part Number Reflects the corresponding node part number from the link topology netlist.
Reserved	21:23	0x0	Reserved. Must be 0x0.

3.7.5.4.3 Get Clock Controller Capabilities Command Errors

The following errors can be returned by the Get Clock Controller Capabilities command:

Table 3-132. Get Clock Controller Capabilities Command Errors

Error Code	Value	Description
OK	0	No error.
ENOENT	2	No such element. - Node handle was not found (for example, Index was higher than the number of handles found for the given context). - Uninitialized chain of PHYs for the given port.
EBUSY	12	PHY/module interface currently busy, retry.
EINVAL	14	Invalid parameters (for example, buffer too small).
NOTINIT	30	LM not initialized.
EFATAL_Initialization	31	LM initialization failure.
EFAIL	255	General error.

3.7.5.5 Set Clock Controller Input Configuration (0x0C62)

This command configures the specified reference clock input of a given Clock Controller (DPLL) node.

3.7.5.5.1 Set Clock Controller Input Configuration Command Inputs

Table 3-133 details the input arguments of the Set Clock Controller Input Configuration command.

Table 3-133. Set Clock Controller Input Configuration Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x0C62	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Input Index	16.[7:0]		Input Index
Update Phase Delay	17.[7]		Update Phase Delay 0b = The <i>Phase Delay</i> field is ignored and the input's current phase delay configuration is kept. 1b = The value of the <i>Phase Delay</i> field is used to update the input's phase delay configuration.
Update Frequency	17.[6]		Update Frequency 0b = The <i>Frequency</i> field is ignored and the input's current frequency configuration is kept. 1b = The value of the <i>Frequency</i> field is used to update the input's frequency configuration.
Reserved	17.[5:0]	0x0	Reserved. Must be 0x0.
Reserved	18.[7]	0b	Reserved. Must be 0b.
ESync Enable	18.[6]		ESync Enable 0b = Disable embedded sync. 1b = Enable embedded sync.
Input Enable	18.[5]		Input Enable 0b = Disable the clock input. 1b = Enable the clock input.
Reserved	18.[4:0]	0x0	Reserved. Must be 0x0.
Reserved	19.[7:0]	0x0	Reserved. Must be 0x0.
Frequency	20:23.[31:0]		Frequency The frequency to be configured with 1 Hz resolution. Note: When <i>Any Frequency</i> is not asserted, the only valid values are 1 Hz and 10 MHz.
Phase Delay	24:27.[31:0]		Phase Delay The phase delay to be configured represented as a signed value with 1 ps resolution
Reserved	28:29.[15:0]	0x0	Reserved. Must be 0x0.
Reserved	30:31.[15:11]	0x0	Reserved. Must be 0x0.

Table 3-133. Set Clock Controller Input Configuration Command Inputs [continued]

Argument	Byte.[Bits]	Value	Description
Node Handle Valid	30:31.[10]		Node Handle Valid 0b = The configuration is applied to the first Clock Controller node in the netlist (i.e. Clock Controller with index 0). 1b = The configuration is applied to the Clock Controller node identified by the <i>Node Handle</i> input.
Node Handle	30:31.[9:0]		Node Handle The handle of the Clock Controller node.

3.7.5.5.2 Set Clock Controller Input Configuration Command Outputs

Table 3-134 details the output arguments of the Set Clock Controller Input Configuration command.

Table 3-134. Set Clock Controller Input Configuration Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x0C62	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.5.5.3, "Set Clock Controller Input Configuration Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Input Index	16.[7:0]		Input Index
Reserved	17:31.[119:0]	0x0	Reserved. Must be 0x0.

3.7.5.5.3 Set Clock Controller Input Configuration Command Errors

The following errors can be returned by the Set Clock Controller Input Configuration command:

Table 3-135. Set Clock Controller Input Configuration Command Errors

Error Code	Value	Description
OK	0	No error.
ENOENT	2	No such element. - Node handle was not found (for example, Index was higher than the number of handles found for the given context). - Uninitialized chain of PHYs for the given port.
EBUSY	12	PHY/module interface currently busy, retry.
EINVAL	14	Invalid parameters (for example, buffer too small).
NOTINIT	30	LM not initialized.
EFATAL_Initialization	31	LM initialization failure.
EFAIL	255	General error.

3.7.5.6 Get Clock Controller Input Configuration (0x0C63)

This command reads the current configuration of the specified reference clock input of a given Clock Controller (DPLL) node.

3.7.5.6.1 Get Clock Controller Input Configuration Command Inputs

Table 3-136 details the input arguments of the Get Clock Controller Input Configuration command.

Table 3-136. Get Clock Controller Input Configuration Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x0C63	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Input Index	16.[7:0]		Input Index
Reserved	17:29.[103:0]	0x0	Reserved. Must be 0x0.
Reserved	30:31.[15:11]	0x0	Reserved. Must be 0x0.
Node Handle Valid	30:31.[10]		Node Handle Valid 0b = The configuration is applied to the first Clock Controller node in the netlist (i.e. Clock Controller with index 0). 1b = The configuration is applied to the Clock Controller node identified by the <i>Node Handle</i> input.
Node Handle	30:31.[9:0]		Node Handle The handle of the Clock Controller node.

3.7.5.6.2 Get Clock Controller Input Configuration Command Outputs

Table 3-137 details the output arguments of the Get Clock Controller Input Configuration command.

Table 3-137. Get Clock Controller Input Configuration Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x0C63	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.5.6.3, "Get Clock Controller Input Configuration Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Input Index	16.[7:0]		Input Index

Table 3-137. Get Clock Controller Input Configuration Command Outputs [continued]

Argument	Byte.[Bits]	Value	Description
ESync Capable	17.[7]		ESync Capable Indicates whether the input can accept a 1PPS sync signal embedded in the clock signal. 0b = The input cannot accept a clock with embedded sync. 1b = The input can be configured to accept a clock with embedded sync.
ESync Fail	17.[6]		ESync Fail Embedded sync error. 0b = No precision frequency monitor failure detected. 1b = The precision frequency monitor failed.
Reserved	17.[5]	0b	Reserved. Must be 0b.
PFM Fail	17.[4]		PFM Fail Precise Frequency Monitor failure. 0b = Embedded sync error detected. 1b = No embedded sync errors were detected.
GST Fail	17.[3]		GST Fail Guard Soak Timer error. 0b = No guard soak timer failure detected. 1b = A guard soak timer failure detected.
CFM Fail	17.[2]		CFM Fail Coarse Frequency Monitor error. 0b = No coarse frequency monitor failure detected. 1b = The coarse frequency monitor failed.
SCM Fail	17.[1]		SCM Fail Single Cycle Monitor error. 0b = No single cycle monitor failure detected. 1b = The single cycle monitor failed.
LOS	17.[0]		LOS Loss of Signal 0b = No loss of signal has been detected. 1b = Loss of signal has been detected.
Reserved	18.[7]	0b	Reserved. Must be 0b.
Recovered Clock	18.[6]		Recovered Clock 0b = The input is not driven by a recovered clock. 1b = The input is driven by a recovered clock.
External Clock	18.[5]		External Clock 0b = The input is not driven by an external clock. 1b = The input is driven by an external clock.
GPS Clock	18.[4]		GPS Clock 0b = The input is not driven by a GPS clock. 1b = The input is driven by a GPS clock.
Reserved	18.[3:1]	000b	Reserved. Must be 000b.
Configurable	18.[0]		Configurable 0b = The input configuration is taken from the link topology netlist and it is not configurable (i.e. it is read only). 1b = The input configuration is configurable.
Any Frequency	19.[7]		Any Frequency 0b = The input only allows for the below listed frequencies. 1b = The input allows for a netlist defined list of frequencies.
Reserved	19.[6:4]	000b	Reserved. Must be 000b.

Table 3-137. Get Clock Controller Input Configuration Command Outputs [continued]

Argument	Byte.[Bits]	Value	Description
10MHz Support	19.[3]		10MHz Support 0b = The input can not accept a 10 MHz clock. 1b = The input can accept a 10 MHz clock.
1PPS Support	19.[2]		1PPS Support 0b = The input can not accept a 1PPS signal. 1b = The input can accept a 1PPS signal.
Reserved	19.[1]	0b	Reserved. Must be 0b.
Phase Delay Support	19.[0]		Phase Delay Support 0b = The input does not support phase delay adjustments. 1b = The input allows for phase delay adjustments.
Frequency	20:23.[31:0]		Frequency The configured frequency configuration with 1 Hz resolution. Note: When Any Frequency is not asserted the only valid values are 1 Hz and 10 MHz.
Phase Delay	24:27.[31:0]		Phase Delay The configured phase delay configuration represented as a signed value with 1 ps resolution.
Reserved	28.[7]	0b	Reserved. Must be 0b.
Esync Enable	28.[6]		Esync Enable 0b = ESsync support is disabled. 1b = ESsync support is enabled.
Input Enable	28.[5]		Input Enable 0b = Clock input is disabled. 1b = Clock input is enabled.
Reserved	28.[4:0]	0x0	Reserved. Must be 0x0.
Reserved	29.[7:0]	0x0	Reserved. Must be 0x0.
Driving Clk Num	30:31.[15:10]		Driving Clock Number The number of the connected clock output on the device pointed to by <i>Driving Node Handle</i> .
Driving Node Handle	30:31.[9:0]		Driving Node Handle Unique identifier of the node in the topology which the I/O is connected to.

3.7.5.6.3 Get Clock Controller Input Configuration Command Errors

The following errors can be returned by the Get Clock Controller Input Configuration command:

Table 3-138. Get Clock Controller Input Configuration Command Errors

Error Code	Value	Description
OK	0	No error.
ENOENT	2	No such element. - Node handle was not found (for example, Index was higher than the number of handles found for the given context). - Uninitialized chain of PHYs for the given port.
EAGAIN	12	PHY/module interface currently busy, retry.
EINVAL	14	Invalid parameters (for example, buffer too small).
NOTINIT	30	LM not initialized.
EFATAL_Initialization	31	LM initialization failure.
EFAIL	255	General error.

3.7.5.7 Set Clock Controller Output Configuration (0x0C64)

This command configures the output frequency, phase compensation, and embedded sync configuration of the specified clock output of a given Clock Controller (DPLL) node.

3.7.5.7.1 Set Clock Controller Output Configuration Command Inputs

Table 3-139 details the input arguments of the Set Clock Controller Output Configuration command.

Table 3-139. Set Clock Controller Output Configuration Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x0C64	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Output Index	16.[7:0]		Output Index
Reserved	17.[7:5]	000b	Reserved. Must be 000b.
Update Src Sel	17.[4]		Update Source Select 0b = The <i>Src Sel</i> field is ignored and the output's current source configuration is kept. 1b = The value of the <i>Src Sel</i> field is used to update the output's source configuration.
Update Phase Delay	17.[3]		Update Phase Delay 0b = The <i>Phase Delay</i> field is ignored and the output's current phase delay configuration is kept. 1b = The value of the <i>Phase Delay</i> field is used to update the output's phase delay configuration.
Update Frequency	17.[2]		Update Frequency 0b = The <i>Frequency</i> field is ignored and the output's current frequency configuration is kept. 1b = The value of the <i>Frequency</i> field is used to update the output's frequency configuration.
ESync Enable	17.[1]		ESync Enable 0b = Disable embedded sync. 1b = Enable embedded sync.
Output Enable	17.[0]		Output Enable 0b = Disable the clock output. 1b = Enable the clock output.
Reserved	18.[7:5]	000b	Reserved. Must be 000b.
Src Sel	18.[4:0]		Source Select The index of the DPLL block to be selected as source for the specified clock output.
Reserved	19.[7:0]	0x0	Reserved. Must be 0x0.
Frequency	20-23.[31:0]		Frequency The frequency to be configured with 1 Hz resolution. Note: When Any Frequency is not asserted the only valid values are 1 Hz and 10 MHz.

Table 3-139. Set Clock Controller Output Configuration Command Inputs [continued]

Argument	Byte.[Bits]	Value	Description
Phase Delay	24:27.[31:0]		Phase Delay The phase delay to be configured represented as a signed value with 1 ps resolution.
Reserved	28:29.[15:0]	0x0	Reserved. Must be 0x0.
Reserved	30:31.[15:11]	0x0	Reserved. Must be 0x0.
Node Handle Valid	30:31.[10]		Node Handle Valid 0b = The configuration is applied to the first Clock Controller node in the netlist (i.e. Clock Controller with index 0). 1b = The configuration is applied to the Clock Controller node identified by the <i>Node Handle</i> input.
Node Handle	30:31.[9:0]		Node Handle The handle of the Clock Controller node.

3.7.5.7.2 Set Clock Controller Output Configuration Command Outputs

Table 3-140 details the output arguments of the Set Clock Controller Output Configuration command.

Table 3-140. Set Clock Controller Output Configuration Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x0C64	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.5.7.3, "Set Clock Controller Output Configuration Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Output Index	16.[7:0]		Output Index
Reserved	17:31.[119:0]	0x0	Reserved. Must be 0x0.

3.7.5.7.3 Set Clock Controller Output Configuration Command Errors

The following errors can be returned by the Set Clock Controller Output Configuration command:

Table 3-141. Set Clock Controller Output Configuration Command Errors

Error Code	Value	Description
OK	0	No error.
ENOENT	2	No such element. - Node handle was not found (for example, Index was higher than the number of handles found for the given context). - Uninitialized chain of PHYs for the given port.
EBUSY	12	PHY/module interface currently busy, retry.
EINVAL	14	Invalid parameters (for example, buffer too small).
NOTINIT	30	LM not initialized.
EFATAL_Initialization	31	LM initialization failure.
EFAIL	255	General error.

3.7.5.8 Get Clock Controller Output Configuration (0x0C65)

This command reads the current frequency, phase compensation, and embedded sync configuration of the specified clock output of a given Clock Controller (DPLL) node.

3.7.5.8.1 Get Clock Controller Output Configuration Command Inputs

Table 3-142 details the input arguments of the Get Clock Controller Output Configuration command.

Table 3-142. Get Clock Controller Output Configuration Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x0C65	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Output Index	16.[7:0]		Output Index
Reserved	17:29.[103:0]	0x0	Reserved. Must be 0x0.
Reserved	30:31.[15:11]	0x0	Reserved. Must be 0x0.
Node Handle Valid	30:31.[10]		Node Handle Valid 0b = The configuration is applied to the first Clock Controller node in the netlist (i.e. Clock Controller with index 0). 1b = The configuration is applied to the Clock Controller node identified by the <i>Node Handle</i> input.
Node Handle	30:31.[9:0]		Node Handle The handle of the Clock Controller node.

3.7.5.8.2 Get Clock Controller Output Configuration Command Outputs

Table 3-143 details the output arguments of the Get Clock Controller Output Configuration command.

Table 3-143. Get Clock Controller Output Configuration Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x0C65	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.5.8.3, "Get Clock Controller Output Configuration Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Output Index	16.[7:0]		Output Index
Reserved	17.[7:3]	0x0	Reserved. Must be 0x0.

Table 3-143. Get Clock Controller Output Configuration Command Outputs [continued]

Argument	Byte.[Bits]	Value	Description
ESync Capable	17.[2]		ESync Capable 0b = The output cannot generate embedded sync. 1b = The output can generate embedded sync.
ESync Enable	17.[1]		ESync Enable 0b = Embedded sync disabled. 1b = Embedded sync enabled.
Output Enable	17.[0]		Output Enable 0b = Clock output disabled. 1b = Clock output enabled.
Mode	18.[7:5]		Mode Reflects the Clock Output Mode configuration from the netlist.
Src Sel	18.[4:0]		Source Select The index of the DPLL block to be selected as source for the specified clock output.
Reserved	19.[7:0]	0x0	Reserved. Must be 0x0.
Frequency	20:23.[31:0]		Frequency The configured output frequency with 1 Hz resolution. Note: When <i>Any Frequency</i> is not asserted, the only valid values are 1 Hz and 10 MHz.
Source Frequency	24:27.[31:0]		Source Frequency The frequency of the clock synthesized the output is derived from with 1 Hz resolution.
Reserved	28:29.[15:0]	0x0	Reserved. Must be 0x0.
Driven Clk Num	30:31.[15:10]		Driven Clock Number The index of the clock input on the node identified by <i>Driven Node Handle</i> the clock output is connected to.
Driven Node Handle	30:31.[9:0]		Driven Node Handle The handle of the node the specified clock output is connected to.

3.7.5.8.3 Get Clock Controller Output Configuration Command Errors

The following errors can be returned by the Get Clock Controller Output Configuration command:

Table 3-144. Get Clock Controller Output Configuration Command Errors

Error Code	Value	Description
OK	0	No error.
ENOENT	2	No such element. - Node handle was not found (for example, Index was higher than the number of handles found for the given context). - Uninitialized chain of PHYs for the given port.
EBUSY	12	PHY/module interface currently busy, retry.
EINVAL	14	Invalid parameters (for example, buffer too small).
NOTINIT	30	LM not initialized.
EFATAL_Initialization	31	LM initialization failure.
EFAIL	255	General error.

3.7.5.9 Get Clock Controller DPLL Status (0x0C66)

This command read the status of the selected DPLL block within the selected CCU node.

3.7.5.9.1 Get Clock Controller DPLL Status Command Inputs

Table 3-145 details the input arguments of the Get Clock Controller DPLL Status command.

Table 3-145. Get Clock Controller DPLL Status Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x0C66	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
DPLL Index	16.[7:0]		DPLL Index
Reserved	17:29.[103:0]	0x0	Reserved. Must be 0x0.
Reserved	30:31.[15:11]	0x0	Reserved. Must be 0x0.
Node Handle Valid	30:31.[10]		Node Handle Valid 0b = The configuration is applied to the first Clock Controller node in the netlist (i.e. Clock Controller with index 0). 1b = The configuration is applied to the Clock Controller node identified by the <i>Node Handle</i> input.
Node Handle	30:31.[9:0]		Node Handle The handle of the Clock Controller node.

3.7.5.9.2 Get Clock Controller DPLL Status Command Outputs

Table 3-146 details the output arguments of the Get Clock Controller DPLL Status command.

Table 3-146. Get Clock Controller DPLL Status Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x0C66	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.5.9.3, "Get Clock Controller DPLL Status Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
DPLL Index	16.[7:0]		DPLL Index
Reserved	17.[7]	0b	Reserved. Must be 0b
REF SW ESync	17.[6]		Reference Switch on ESync Fail 0b = The DPLL does not switch references on embedded sync failures. 1b = When configured to automatic mode, the DPLL switches references on embedded sync failures.

Table 3-146. Get Clock Controller DPLL Status Command Outputs [continued]

Argument	Byte.[Bits]	Value	Description
Reserved	17.[5]	0b	Reserved. Must be 0b
REF SW PFM	17.[4]		Reference Switch on Precision Frequency Monitor Fail 0b = The DPLL does not switch references on precision frequency monitor failures. 1b = When configured to automatic mode, the DPLL switches references on precision frequency monitor failures.
REF SW GST	17.[3]		Reference Switch on Guard Soak Timer Fail 0b = The DPLL does not switch references on guard soak timer failures. 1b = When configured to automatic mode, the DPLL switches references on guard soak timer failures.
REF SW CFM	17.[2]		Reference Switch on Coarse Frequency Monitor Fail 0b = The DPLL does not switch references on coarse frequency monitor failures. 1b = When configured to automatic mode, the DPLL switches references on coarse frequency monitor failures.
REF SW SCM	17.[1]		Reference Switch on Single Cycle Monitor Fail 0b = The DPLL does not switch references on single cycle monitor failures. 1b = When configured to automatic mode, the DPLL switches references on single cycle monitor failures.
REF SW LOS	17.[0]		Reference Switch on Loss of Signal 0b = The DPLL does not switch references on loss of signal detected. 1b = When configured to automatic mode, the DPLL switches references on loss of signal detected.
PSL Hit	18.[7]		Phase Slope Limit Hit 0b = The phase change of the DPLL block has not been limited. 1b = The phase change of the DPLL block has been limited by the phase slope limiter.
Reserved	18.[6]	0b	Reserved. Must be 0b
FL Hit	18.[5]		Frequency Limit Hit 0b = The frequency of the selected reference is within of the allowed pull-in/hold-in range. 1b = The frequency of the selected reference is outside of the allowed pull-in/hold-in range.
Reserved	18.[4:3]	00b	Reserved. Must be 00b
HO Ready	18.[2]		Holdover Ready 0b = The DPLL is not ready for holdover. 1b = The DPLL is ready for holdover
HO	18.[1]		Holdover 0b = The DPLL is not in holdover. 1b = The DPLL is in holdover.
Lock	18.[0]		Lock 0b = The DPLL is not locked. 1b = The DPLL is locked to the selected reference.
Mode	19.[7:5]		Mode 000b = FreeRun 001b = ForceHoldover 010b = ForceLock 011b = Auto 100b = Numerically Controlled Oscillator All other values are reserved.

Table 3-146. Get Clock Controller DPLL Status Command Outputs [continued]

Argument	Byte.[Bits]	Value	Description
Clk Ref Sel	19.[4:0]		Clock Reference Select The index of the currently selected reference clock input
Reserved	20:23.[31:16]	0x0	Reserved. Must be 0x0.
Phase Offset High	20:23.[15:0]		Phase Offset High Measured Phase Err[47:32] The measured phase error relative to the currently selected reference clock input with a 1 ps resolution.
Phase Offset Low	24:27.[31:0]		Phase Offset Low Measured Phase Err[31:0] The measured phase error relative to the currently selected reference clock input with a 1 ps resolution.
Reserved	28.[7:4]	0x0	Reserved. Must be 0x0.
EEC Mode	28.[3:0]		EEC Mode 0xA = QL-EEC2/QL-ST3 0xB = QL-EEC1/QL-SEC 0xF = QL-DUS All other values are reserved.
Reserved	29:31.[23:0]	0x0	Reserved. Must be 0x0.

3.7.5.9.3 Get Clock Controller DPLL Status Command Errors

The following errors can be returned by the Get Clock Controller DPLL Status command:

Table 3-147. Get Clock Controller DPLL Status Command Errors

Error Code	Value	Description
OK	0	No error.
ENOENT	2	No such element. - Node handle was not found (for example, Index was higher than the number of handles found for the given context). - Uninitialized chain of PHYs for the given port.
EBUSY	12	PHY/module interface currently busy, retry.
EINVAL	14	Invalid parameters (for example, buffer too small).
NOTINIT	30	LM not initialized.
EFATAL_Initialization	31	LM initialization failure.
EFAIL	255	General error.

3.7.5.10 Set Clock Controller DPLL Configuration (0x0C67)

This command configures the selected DPLL block within the selected CCU node.

3.7.5.10.1 Set Clock Controller DPLL Configuration Command Inputs

Table 3-148 details the input arguments of the Set Clock Controller DPLL Configuration command.

Table 3-148. Set Clock Controller DPLL Configuration Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x0C67	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
DPLL Index	16.[7:0]		DPLL Index
Reserved	17.[7]	0b	Reserved. Must be 0b
REF SW ESync	17.[6]		Reference Switch on ESync Fail 0b = The DPLL does not switch references on embedded sync failures. 1b = When configured to automatic mode, the DPLL switches references on embedded sync failures.
Fast Lock Enable	17.[5]		Fast Lock Enable Increases DPLL loop bandwidth to allow faster lock to 1PPS signals. 0b = Fast Lock disabled. 1b = Fast Lock enabled.
REF SW PFM	17.[4]		Reference Switch on Precision Frequency Monitor Fail 0b = The DPLL does not switch references on precision frequency monitor failures. 1b = When configured to automatic mode, the DPLL switches references on precision frequency monitor failures.
REF SW GST	17.[3]		Reference Switch on Guard Soak Timer Fail 0b = The DPLL does not switch references on guard soak timer failures. 1b = When configured to automatic mode, the DPLL switches references on guard soak timer failures.
REF SW CFM	17.[2]		Reference Switch on Coarse Frequency Monitor Fail 0b = The DPLL does not switch references on coarse frequency monitor failures. 1b = When configured to automatic mode, the DPLL switches references on coarse frequency monitor failures.
REF SW SCM	17.[1]		Reference Switch on Single Cycle Monitor Fail 0b = The DPLL does not switch references on single cycle monitor failures. 1b = When configured to automatic mode, the DPLL switches references on single cycle monitor failures.
REF SW LOS	17.[0]		Reference Switch on Loss of Signal 0b = The DPLL does not switch references on loss of signal detected. 1b = When configured to automatic mode, the DPLL switches references on loss of signal detected.
Reserved	18.[7:0]	0x0	Reserved. Must be 0x0.

Table 3-148. Set Clock Controller DPLL Configuration Command Inputs [continued]

Argument	Byte.[Bits]	Value	Description
Mode	19.[7:5]		Mode 000b = FreeRun 001b = ForceHoldover 010b = ForceLock 011b = Auto 100b = Numerically Controlled Oscillator All other values are reserved.
Clk Ref Sel	19.[4:0]		Clock Reference Select The index of the currently selected reference clock input
Reserved	20:27.[63:0]	0x0	Reserved. Must be 0x0.
Reserved	28.[7:4]	0x0	Reserved. Must be 0x0.
EEC Mode	28.[3:0]		EEC Mode 0xA = QL-EEC2/QL-ST3 0xB = QL-EEC1/QL-SEC 0xF = QL-DUS All other values are reserved.
Reserved	29.[7:0]	0x0	Reserved. Must be 0x0.
Reserved	30:31.[15:11]	0x0	Reserved. Must be 0x0.
Node Handle Valid	30:31.[10]		Node Handle Valid 0b = The configuration is applied to the first Clock Controller node in the netlist (i.e. Clock Controller with index 0). 1b = The configuration is applied to the Clock Controller node identified by the <i>Node Handle</i> input.
Node Handle	30:31.[9:0]		Node Handle The handle of the Clock Controller node.

3.7.5.10.2 Set Clock Controller DPLL Configuration Command Outputs

Table 3-149 details the output arguments of the Set Clock Controller DPLL Configuration command.

Table 3-149. Set Clock Controller DPLL Configuration Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x0C67	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.5.10.3, "Set Clock Controller DPLL Configuration Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
DPLL Index	16.[7:0]		DPLL Index
Reserved	17:31.[119:0]	0x0	Reserved. Must be 0x0.

3.7.5.10.3 Set Clock Controller DPLL Configuration Command Errors

The following errors can be returned by the Set Clock Controller DPLL Configuration command:

Table 3-150. Set Clock Controller DPLL Configuration Command Errors

Error Code	Value	Description
OK	0	No error.
ENOENT	2	No such element. - Node handle was not found (for example, Index was higher than the number of handles found for the given context). - Uninitialized chain of PHYs for the given port.
EBUSY	12	PHY/module interface currently busy, retry.
EINVAL	14	Invalid parameters (for example, buffer too small).
NOTINIT	30	LM not initialized.
EFATAL_Initialization	31	LM initialization failure.
EFAIL	255	General error.

3.7.5.11 Set Clock Controller Reference Priority (0x0C68)

This command configures the priority of the selected Input Index within a given DPLL block of CCU node.

3.7.5.11.1 Set Clock Controller Reference Priority Command Inputs

Table 3-151 details the input arguments of the Set Clock Controller Reference Priority command.

Table 3-151. Set Clock Controller Reference Priority Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x0C68	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
DPLL Index	16.[7:0]		DPLL Index
Ref Index	17.[7:0]		Reference Index The index of the reference input.
Ref Priority	18.[7:0]		Reference Clock Priority The priority to be assigned to the reference clock input identified by <i>Ref Index</i> . A value of 0 denotes the highest priority. Note: A value of 0xFF results in removing the specified reference clock input from the list of allowed reference clock options for the specified DPLL block.
Reserved	19:29.[87:0]	0x0	Reserved. Must be 0x0.
Reserved	30:31.[15:11]	0x0	Reserved. Must be 0x0.
Node Handle Valid	30:31.[10]		Node Handle Valid 0b = The configuration is applied to the first Clock Controller node in the netlist (i.e. Clock Controller with index 0). 1b = The configuration is applied to the Clock Controller node identified by the <i>Node Handle</i> input.
Node Handle	30:31.[9:0]		Node Handle The handle of the Clock Controller node.

3.7.5.11.2 Set Clock Controller Reference Priority Command Outputs

Table 3-152 details the output arguments of the Set Clock Controller Reference Priority command.

Table 3-152. Set Clock Controller Reference Priority Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x0C68	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.5.11.3, "Set Clock Controller Reference Priority Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
DPLL Index	16.[7:0]		DPLL Index
Ref Index	17.[7:0]		Reference Index The index of the reference input.
Ref Priority	18.[7:0]		Reference Clock Priority The priority to be assigned to the reference clock input identified by <i>Ref Index</i> . A value of 0 denotes the highest priority. Note: A value of 0xFF results in removing the specified reference clock input from the list of allowed reference clock options for the specified DPLL block.
Reserved	19:31.[103:0]	0x0	Reserved. Must be 0x0.

3.7.5.11.3 Set Clock Controller Reference Priority Command Errors

The following errors can be returned by the Set Clock Controller Reference Priority command:

Table 3-153. Set Clock Controller Reference Priority Command Errors

Error Code	Value	Description
OK	0	No error.
ENOENT	2	No such element. - Node handle was not found (for example, Index was higher than the number of handles found for the given context). - Uninitialized chain of PHYs for the given port.
EBUSY	12	PHY/module interface currently busy, retry.
EINVAL	14	Invalid parameters (for example, buffer too small).
NOTINIT	30	LM not initialized.
EFATAL_Initialization	31	LM initialization failure.
EFAIL	255	General error.

3.7.5.12 Get Clock Controller Reference Priority (0x0C69)

This command reads the currently configured priority of the selected reference clock for a given DPLL block within a given Clock Controller (DPLL) node.

3.7.5.12.1 Get Clock Controller Reference Priority Command Inputs

Table 3-154 details the input arguments of the Get Clock Controller Reference Priority command.

Table 3-154. Get Clock Controller Reference Priority Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x0C69	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
DPLL Index	16.[7:0]		DPLL Index
Ref Index	17.[7:0]		Reference Index The index of the reference input.
Reserved	18:29.[95:0]	0x0	Reserved. Must be 0x0.
Reserved	30:31.[15:11]	0x0	Reserved. Must be 0x0.
Node Handle Valid	30:31.[10]		Node Handle Valid 0b = The configuration is applied to the first Clock Controller node in the netlist (i.e. Clock Controller with index 0). 1b = The configuration is applied to the Clock Controller node identified by the <i>Node Handle</i> input.
Node Handle	30:31.[9:0]		Node Handle The handle of the Clock Controller node.

3.7.5.12.2 Get Clock Controller Reference Priority Command Outputs

Table 3-155 details the output arguments of the Get Clock Controller Reference Priority command.

Table 3-155. Get Clock Controller Reference Priority Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x0C69	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.5.12.3, "Get Clock Controller Reference Priority Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
DPLL Index	16.[7:0]		DPLL Index

Table 3-155. Get Clock Controller Reference Priority Command Outputs [continued]

Argument	Byte.[Bits]	Value	Description
Ref Index	17.[7:0]		Reference Index The index of the reference input.
Ref Priority	18.[7:0]		Reference Clock Priority The priority to be assigned to the reference clock input identified by <i>Ref Index</i> . A value of 0 denotes the highest priority. Note: A value of 0xFF results in removing the specified reference clock input from the list of allowed reference clock options for the specified DPLL block.
Reserved	19:31.[103:0]	0x0	Reserved. Must be 0x0.

3.7.5.12.3 Get Clock Controller Reference Priority Command Errors

The following errors can be returned by the Get Clock Controller Reference Priority command:

Table 3-156. Get Clock Controller Reference Priority Command Errors

Error Code	Value	Description
OK	0	No error.
ENOENT	2	No such element. - Node handle was not found (for example, Index was higher than the number of handles found for the given context). - Uninitialized chain of PHYs for the given port.
EBUSY	12	PHY/module interface currently busy, retry.
EINVAL	14	Invalid parameters (for example, buffer too small).
NOTINIT	30	LM not initialized.
EFATAL_Initialization	31	LM initialization failure.
EFAIL	255	General error.

3.7.5.13 Get Clock Controller Info (0x0C6A)

This command retrieves information about the CCU. If the parameter is unsupported, it should contain 0xFF for 8-bit values or 0xFFFF FFFF for 32-bit values.

3.7.5.13.1 Get Clock Controller Info Command Inputs

Table 3-157 details the input arguments of the Get Clock Controller Info command.

Table 3-157. Get Clock Controller Info Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x0C6A	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:29.[111:0]	0x0	Reserved. Must be 0x0.
Reserved	30:31.[15:11]	0x0	Reserved. Must be 0x0.
Node Handle Valid	30:31.[10]		Node Handle Valid 0b = The configuration is applied to the first Clock Controller node in the netlist (i.e. Clock Controller with index 0). 1b = The configuration is applied to the Clock Controller node identified by the <i>Node Handle</i> input.
Node Handle	30:31.[9:0]		Node Handle The handle of the Clock Controller node.

3.7.5.13.2 Get Clock Controller Info Outputs

Table 3-158 details the output arguments of the Get Clock Controller Info command.

Table 3-158. Get Clock Controller Info Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x0C6A	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.5.13.3, "Get Clock Controller Info Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Manufacturer's Part Number	16:19.[31:0]		Manufacturer's Part Number Reflects the manufacturer's part number or device ID retrieved from the device.
Custom Configuration Version	20:23.[31:0]		Custom Configuration Version Reflects the Custom Configuration Version retrieved from the device. Note: Returns 0xFFFFFFFF if the Configuration ID is not supported.

Table 3-158. Get Clock Controller Info Command Outputs [continued]

Argument	Byte.[Bits]	Value	Description
FW Version	24:27.[31:0]		Firmware Version Reflects the firmware version retrieved from the device.
Node Part Number	28.[7:0]		Node Part Number Reflects the corresponding node part number from the link topology netlist.
Device Revision	29.[7:0]		Device Revision Reflects the silicon stepping version retrieved from the device.
Reserved	30:31.[15:0]	0x0	Reserved. Must be 0x0.

3.7.5.13.3 Get Clock Controller Info Command Errors

The following errors can be returned by the Get Clock Controller Info command:

Table 3-159. Get Clock Controller Info Command Errors

Error Code	Value	Description
OK	0	No error.
EBUSY	12	PHY/module interface currently busy, retry.
EINVAL	14	Invalid parameters.

3.7.6 Topology Flows

3.7.6.1 Overview

The Link Manager's Topology Engine maintains the data structure describing the board topology, also referred to as link topology. This structure is initialized after startup and updated following all hot-plug events.

After reset, first the LOM topology is detected. This is followed by initialization of the topology nodes described by the LOM topology.

Once the sideband devices part of the LOM topology has been initialized, the presence of potential mezzanine cards is determined. For each mezzanine card present, the mezzanine topology is detected and merged with the already discovered topology. The topology nodes of the mezzanine card are initialized before moving on to the next mezzanine card.

Note: In the E610, the mezzanine card does not exist.

3.7.6.2 Detecting the LOM/NIC Topology

The topology of the motherboard (LOM) or network interface card (NIC) is detected by the Topology Engine reading the topology netlist information from a binary file located on the file system from which the IMC is booted. In the process, all PHY and sideband connections are detected.

For PHY nodes, the netlist can capture several Port Options.

- For PHY nodes with the *Innermost* bit set, the active Port Option is loaded to RAM. If no Port Option is marked as active, the first one (the one with highest priority) is selected as active.
- For PHY nodes with the *Innermost* bit clear, first the active Port Option is selected and then loaded to RAM. The active Port Option for these PHY nodes is determined by matching its Port Options to the active Port Option of the connected *Innermost* PHY.

The Port Options in the netlist are captured in order of decreasing priority. The matching option with the highest priority is selected.

For LED and Temperature Sensor nodes, the netlist captures separate configuration for each Port Option of their Parent PHY node. Only the configuration associated with the Parent PHY's active Port Option is loaded to RAM.

For all other nodes all corresponding netlist configuration is loaded to RAM.

The detected devices are initialized through the Link Manager repeatedly calling the INIT_BOARD DNL script.

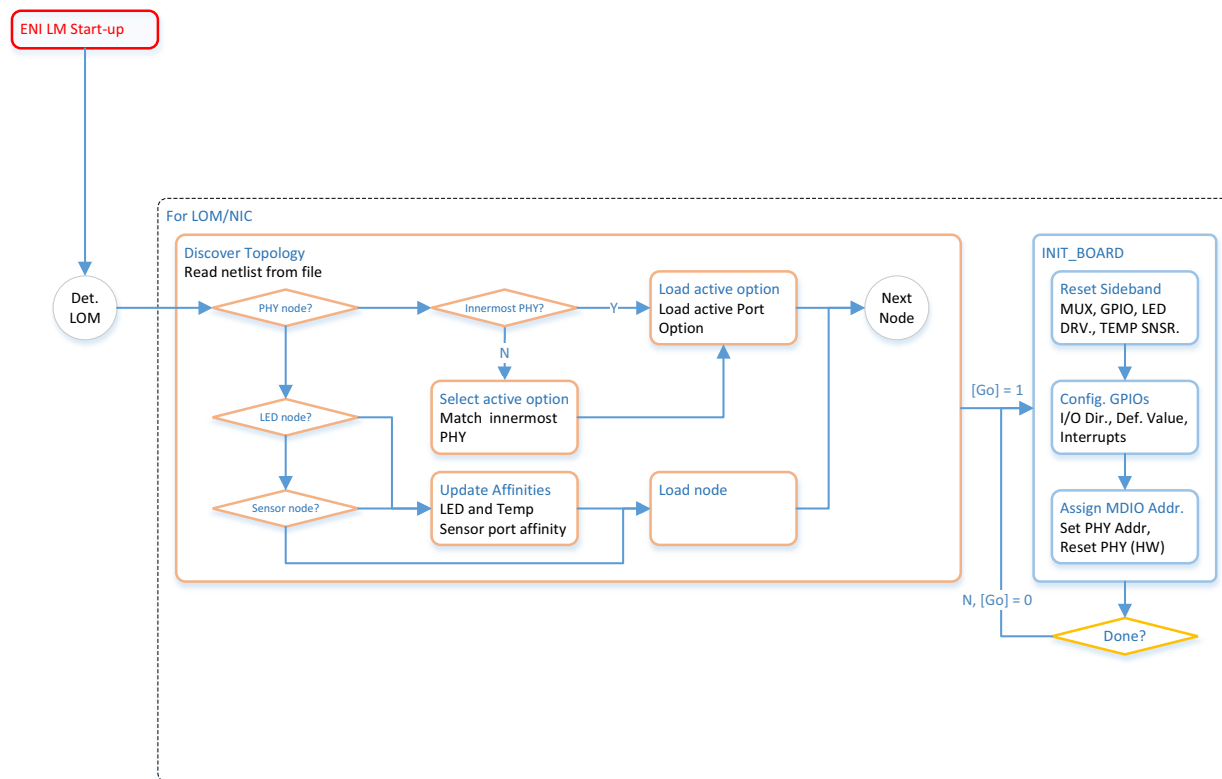


Figure 3-31. LOM/NIC Topology Detection Flow

Once the topology netlist has been updated, the detected devices on the mezzanine card are initialized through the Link Manager repeatedly calling the INIT_BOARD DNL script.

3.7.6.3 Topology Initialization

The board topology initialization is performed by the Link Manager repeatedly calling the INIT_BOARD DNL script. The first call is placed with the *GO* bit set, followed by calls with the *GO* bit cleared until the script returns with the *DONE* bit set.

All boards must be initialized after discovery.

The motherboard must be initialized before presence of mezzanine cards can be detected and mezzanine cards can be initialized.

The repeated calls to INIT_BOARD facilitate the state machine shown in [Figure 3-32](#).

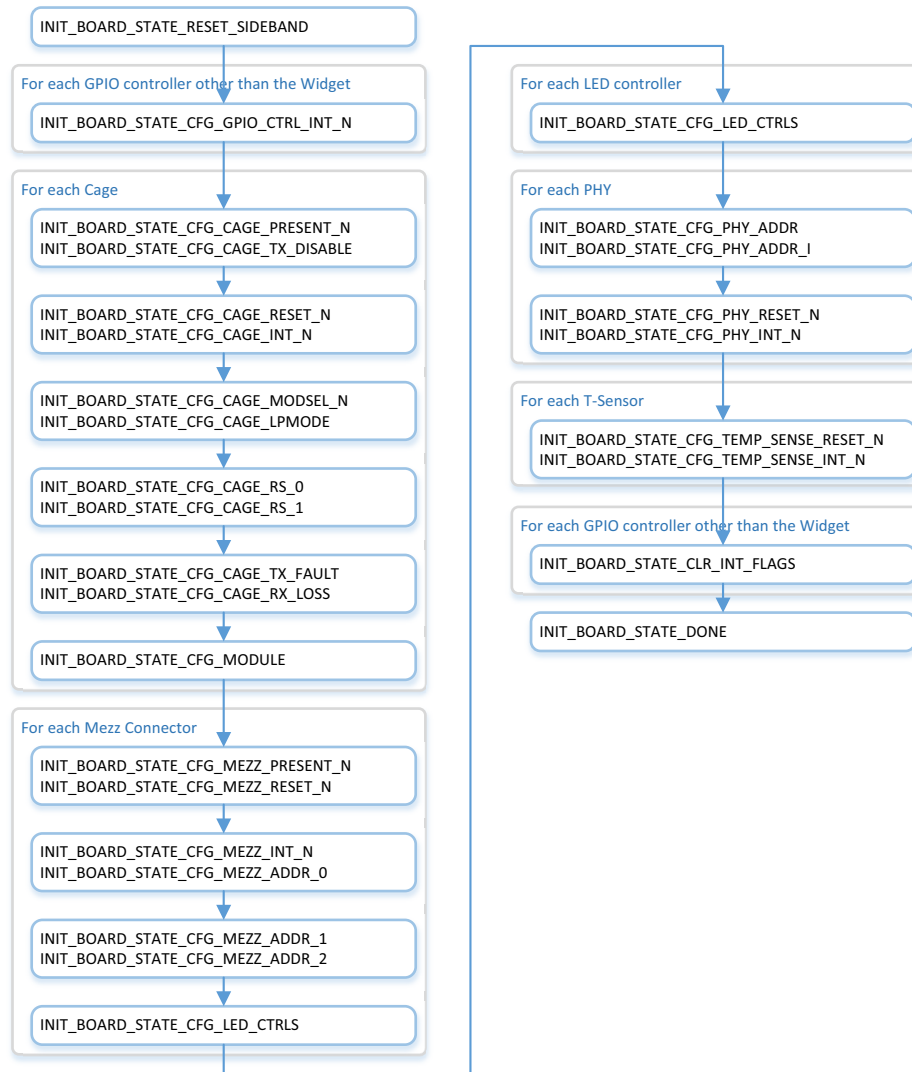


Figure 3-32. Board Initialization Flow

Depending on the nodes defined in the netlist, the state machine might pass through some of these states several times or skip them entirely.

The initialization begins with resetting all sideband devices on the board by asserting and de-asserting their shared reset signal. This shared reset signal is identified as the reset signal of any of the board's GPIO controller, mux controller, LED controller or Temperature sensor nodes.

Next, the flow continues with configuring the GPIOs connected to the INT_N signal of all GPIO controllers as an input generating interrupts based on netlist settings.

Once the interrupt signals of all GPIO controllers have been addressed, the initialization flow continues configuring the GPIOs connected to the PRESENT_N, TX_DISABLE, RESET_N, INT_N, MODSEL_N, LPMODE, RS_0, or RS_1 signal of all cage nodes with I/O direction defined by their I/O Function and netlist settings.

In the INIT_BOARD_STATE_CFG_CAGE_MODULE, the script determines whether a module is present or not and if a module is detected calls SFF_MODULE_INIT to reset the module, assert TX_DISABLE, and set low power mode.

Once the signals of all cages and the modules present in them have been addressed, the initialization flow continues configuring the GPIOs connected to the PRESENT_N, RESET_N, INT_N, ADDR_0, ADDR_1, and ADDR_2 signals of all mezzanine connector nodes with I/O direction defined by their I/O Function and netlist settings.

Next all LED controllers are initialized by resetting them to their default configuration via the RESET_LED_CONTROLLER script.

Once all LED controllers have been addressed, the initialization flow continues with updating the bus address of all PHY nodes with relative node address type based on the board number and the original bus address of the node. All GPIOs driving the ADDR_[IO Index] pins of each of these PHYs are configured to reflect the corresponding bit location of the binary representation of the updated bus address of the PHY node.

This is followed by configuring the GPIOs connected to the RESET_N, INT_N signals of all PHY nodes with I/O direction defined by their I/O Function and netlist settings.

Next the initialization flow continues with updating the bus address of all temperature sensor nodes with relative node address type based on the board number and the original bus address of the node.

This is followed by configuring the GPIOs connected to the RESET_N, INT_N signals of all temperature sensor nodes with I/O direction defined by their I/O Function and netlist settings.

Once the signals of all temperature sensors have been addressed, the initialization flow continues with clearing all potential interrupt flags by calling GET_INT_FLAGS with the CLEAR_FLAGS flag set for all GPIO controllers.

Once all board topologies have been initialized, the Link Manager collects information about all PHYs and initiates loading of PHY firmware as needed:

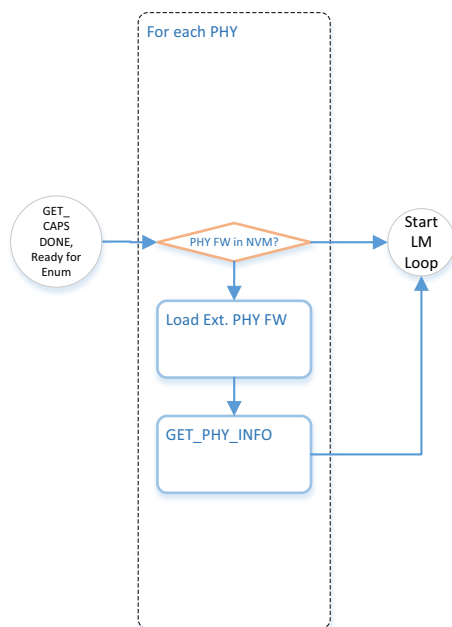


Figure 3-33. PHY Firmware Load

3.7.6.4 Topology Port Options

The link topology netlist can capture multiple port configurations, referred to as port options, for any PHY node within the link topology. When the topology is detected, only the port option marked as active is loaded from the topology binary file.

Once all board topologies have been initialized, the rest of the data-path is configured to align with the number of ports defined by the active port option based on the Global Configuration ID selected by active port option of the innermost PHY.

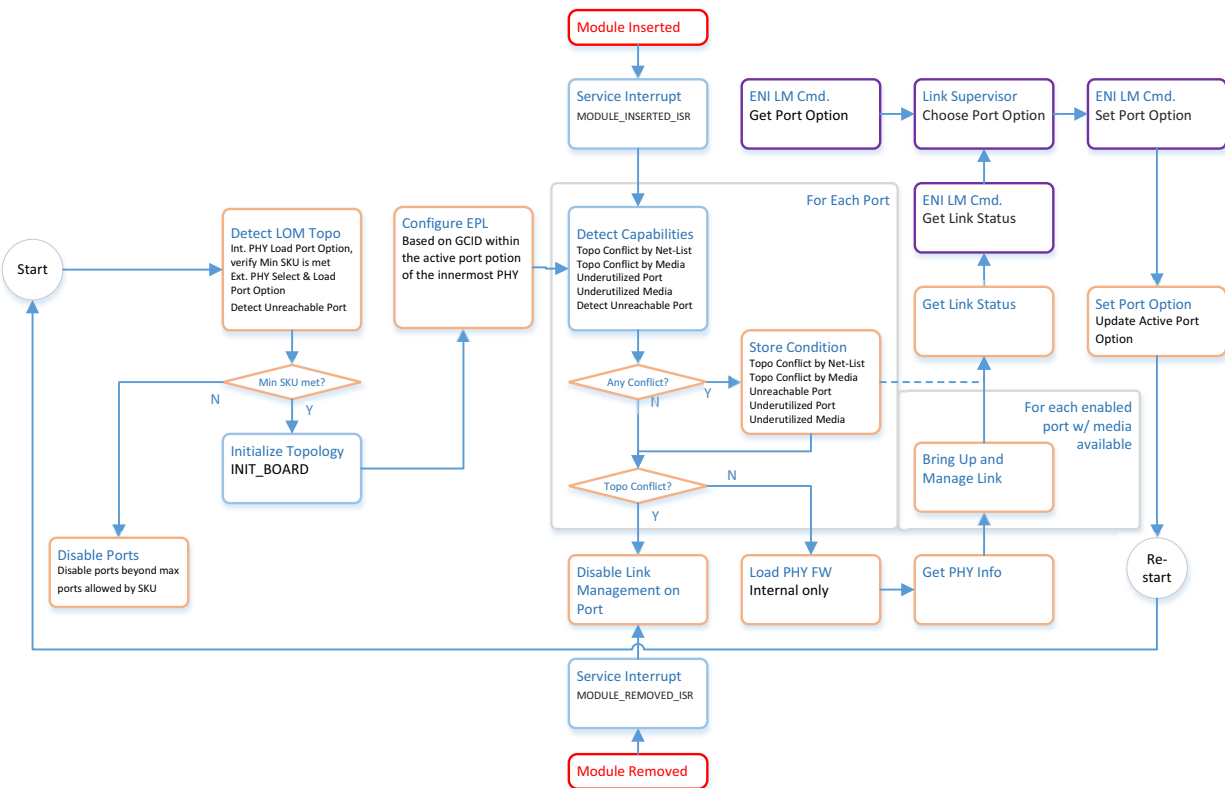


Figure 3-34. Port Option Selection Flow

3.7.6.4.1 Detecting Conflicts

Some link topologies can result certain conflicts impacting the behavior of the Link Manager.

The possible types of conflicts are:

- Media Conflicts caused by a mismatch of the active Port Option of the innermost PHY and the connected media.
- Topology Conflict caused by Media - Lane[0] of at least one port maps into a non-implemented lane of the media.
- Unreachable Port - Lane[0] of an outermost PMD doesn't have a connection to Lane[0] of any PMD on the driving node.

- Underutilized Port - At least one lane of the port maps into a non implemented lane of the media, or the port is wider than the media (ex. breakout cable).
- Underutilized Media - At least one lane of the media is not connected to any lane of any PMD on the driving node.
- Topology Conflicts caused by a mismatch of the active Port Option of the Innermost PHY and the active Port Option of a connected mezzanine card.
- Topology Conflict caused by netlist - The corresponding PMD on the outermost PHY (PHY[0]) has the Outermost bit not set or the capabilities resulting from intersection of all link segments and fuses is null.

These possible conflicts are identified during detecting the capabilities of the enabled ports. Their presence is determined by the GET_CAPABILITIES0/1 DNL scripts described in PHY Capability Detection after the topology has been discovered and for ports with a cage associated every time after a module is inserted. To minimize the number of IC transactions, all relevant module information is cached to the netlist and the resulting conditions are stored for later consumption.

3.7.6.4.2 Resolving Conflicts: Changing Port Options

If conflicts have been detected, they might need to be resolved before link can be established on the affected ports. This can be accomplished by selecting the appropriate port option.

To select a port option first the list of available port options must be retrieved. This can be done via the Get Port Options link topology configuration command.

Applying one of the available options can be triggered via the Set Port Option configuration command.

Applying a new port option is disruptive for links on both quads. It results in a request for LINKR and the Link Manager re-initializes.

Note: In the E610, both Get Port Options and Set Port Option are not implemented, as there is only one port option.

3.7.6.5 Topology Interrupt Sources and Causes

The link topology related interrupts are captured by the Link Manager. Their cause must be identified and then serviced.

The source and cause of interrupts generated within the link topology is identified by the Link Manager recursively calling the GET_INT_CAUSE DNL script for all nodes with an asserted interrupt flag starting at the root of the interrupt tree. Only the interrupt flags associated with the GPIOs allocated to the Link Manager are considered: GPIO[8, 9, 20:27] (i.e. MISCIO[30, 20, 41:48] or PORTA[8, 9, 20:27]).

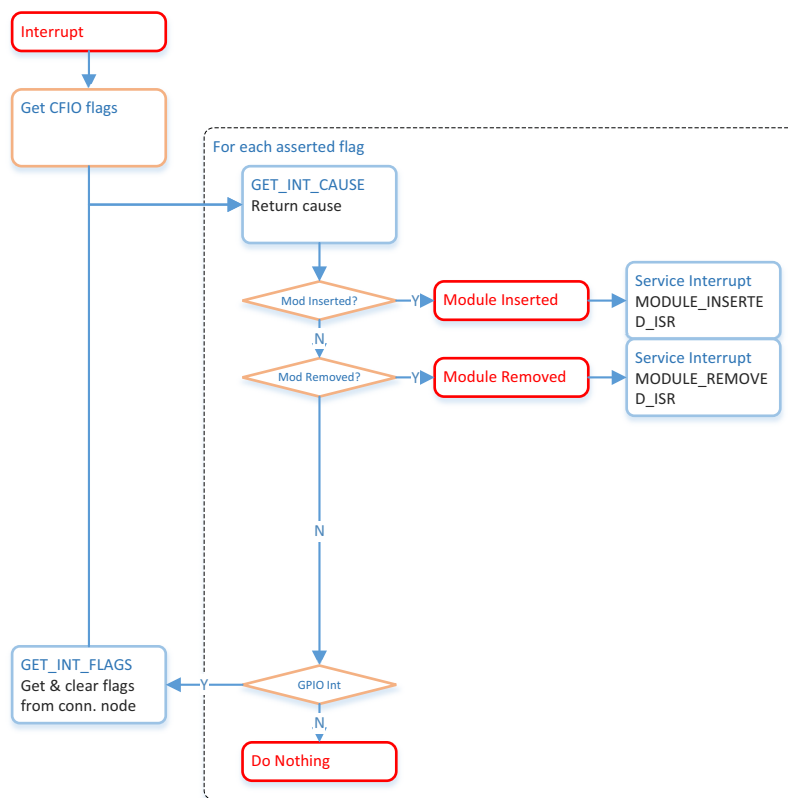


Figure 3-35. Interrupt Cause Flow

The returned arguments indicate all flags currently set and the cause of the selected flag. By iterating through all asserted flags, the Link Manager retrieves the cause of all interrupts generated.

If the cause is identified to be a generic GPIO, the interrupt was generated by a downstream node. Following the I/O Connection defined by the netlist the Link Manager identifies the node generating that interrupt and recursively retrieves and handles the interrupt cause for all asserted flags. Based on the interrupt cause identified the, Link Manager calls the appropriate DNL activities to service the interrupt.

Note: Only hot-plug (module inserted, module removed, mezzanine inserted, and mezzanine removed) events are handled through interrupts.

3.7.6.6 Hot-Plug Events

Hot-plug events include insertion or removal of pluggable modules and mezzanine cards. They are detected through interrupts generated by the PRESENT_N pin of Cage or Mezzanine Connector nodes.

In case of SFP+/SFP28 cages, this I/O is referring to the MOD_ABS pin.

3.7.6.6.1 Module Inserted

When a new module is inserted, the change in state of the presence pin generates an interrupt. Once the Link Manager determines the cause of the interrupt to be the insertion of a new pluggable module, it repeatedly calls the MODULE_INSERTED_ISR DNL script to service the interrupt. The first call is placed with the GO bit set, followed by calls with the GO bit cleared until the script returns with the *DONE* bit set.

The interrupt service routine resets and transitions the module to a low power state and disables Transmit where applicable.

As shown in Figure 3-34 and Figure 3-36, the Link Manager retrieves and saves the capabilities of the affected ports and continues to identify and handle potential conflicts caused by the new module.

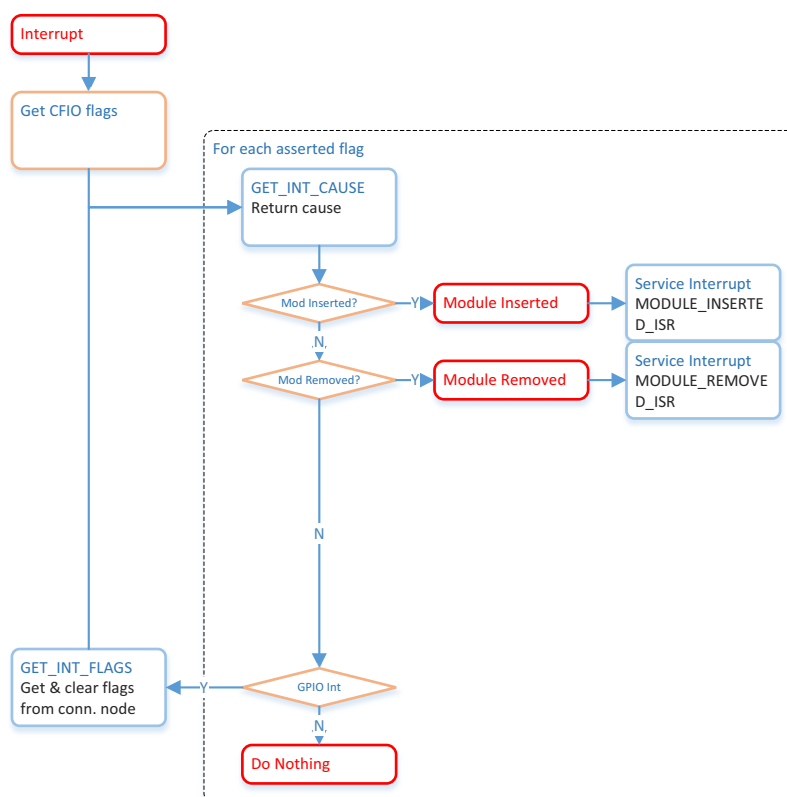


Figure 3-36. Module Inserted Flow

The event generates an LSE corresponding to the change in media availability.

3.7.6.6.2 Module Removed

When a module is removed, the change in state of the presence pin generates an interrupt. Once the Link Manager determines the cause of the interrupt to be the removal of a pluggable module, it calls the MODULE_REMOVED_ISR DNL script to service the interrupt.

The interrupt service routine will clean up all saved module information.

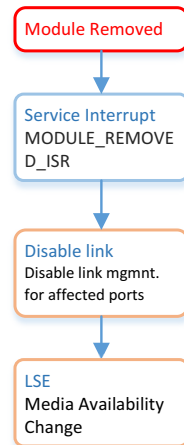


Figure 3-37. Module Removed Flow

The event generates an LSE corresponding to the change in media availability.

In the absence of a module, when requesting capabilities with the media included, GET_CAPABILITIES_0, GET_CAPABILITIES_1, and Get PHY Capabilities return all possible options supported by the cage and current port configuration.

3.7.6.7 Temperature Sensors

The netlist may define several thermal sensors. After initialized at the end of the board initialization process, the state of these sensors is monitored through polling, reading their value every 1 second.

The readings from the SFP module are compared against the thermal thresholds defined in the module.

The readings from other nodes are compared against the thermal thresholds defined in the netlist.

If the temperature level raises above the Fatal Threshold, the Link Manager disables link on the affected ports through the SET_LINK_DOWN DNL script.

Every time the temperature level raises above any of the defined thresholds, the Link Manager triggers a corresponding PLDM message.

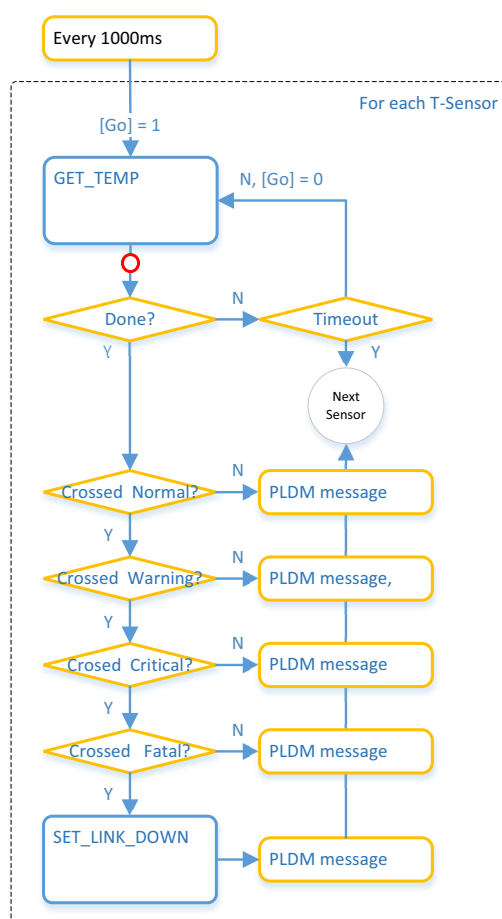


Figure 3-38. Temperature Sensor Flow

3.7.7 Topology Commands

Most components of the link topology are mainly accessed by the Link Manager, either directly or through Dreadnought Lake scripts. Sometimes however, there is a need for the software driver or software tools to access the link topology components. As link topology components are owned by the Link Manager, any such access should be coordinated and conducted thorough link topology commands, allowing the Link Manager to synchronize and control the required access.

Note: Logical port number in topology commands - The topology commands might use a logical port number to identify the topology device they access. Since the topology netlist starts with the innermost PHY and its ports, the Link Manager would translate the logical port into physical MAC port and would identify the PHY port of the innermost PHY that is connected to the physical MAC port. When the physical function owns a single port, there is no need to use the logical port number field.

Note: All reserved fields within the command arguments must be 0.

3.7.7.1 Topology Commands Summary

Table 3-160. Link Topology Commands

Command	Opcode	Description	Section Reference
Get Link Topology Handle	0x06E0	Get the handle for a link topology node.	Section 3.7.7.2
Get Link Topology Pin	0x06E1	Get the handle for a link topology pin.	Section 3.7.7.3
Read I2C	0x06E2	Read register from I ² C attached device.	Section 3.7.7.4
Write I2C	0x06E3	Write register of I ² C attached device.	Section 3.7.7.5
Read MDIO	0x06E4	Read register from MDIO device	Section 3.7.7.6
Write MDIO	0x06E5	Write register to MDIO device	Section 3.7.7.7
Set GPIO By Function	0x06E6	Set value to GPIO signal referenced by its function.	Section 3.7.7.8
Get GPIO By Function	0x06E7	Get value from GPIO signal referenced by its function.	Section 3.7.7.9
Set LED	0x06E8	Set the blink option and color of a LED.	Section 3.7.7.10
Set Port Identification LED	0x06E9	Set the LED used for identification of this port.	Section 3.7.7.11
Set GPIO	0x06EC	Set value to GPIO signal	Section 3.7.7.12
Get GPIO	0x06ED	Get value from GPIO signal.	Section 3.7.7.13
Read/Write SFF/CMIS Module	0x06EE	Read register from I ² C attached SFF/CMIS defined pluggable module.	Section 3.7.7.14
Program Topology Device NVM	0x06F2	Triggers the write of the topology device NVM.	Section 3.7.7.15
Read Topology Device NVM	0x06F3	Read the topology device NVM	Section 3.7.7.16

3.7.7.2 Get Link Topology Handle (0x06E0)

The Get Link Topology Handle command is used to retrieve the node handle for a topology device. Most of the link topology commands can reference the node according to specific context (i.e. it can search for a node at a specific "Board", "Port" or "Node" context or at the global context). In this case, the software should provide the port number, the node type and node index to access the node.

A reference node should be provided, when the context of the search is "Node". However, it is sometimes more efficient to retrieve first the node handle and use it for a sequence of operations to access this node. In those case, after retrieving the node using the Get Link Topology Handle configuration command, the subsequent commands can be called directly with the node handle, flagging that a specific node handle is directly provided.

The node type and node index are used to identify the exact topology device needed. The node index is needed when the specific node type is used more than once in a specific topology context.

3.7.7.2.1 Get Link Topology Handle Command Inputs

Table 3-161 details the input arguments of the Get Link Topology Handle command.

Table 3-161. Get Link Topology Handle Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x06E0	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Logical Port Number	16.[7:0]		Logical Port Number This field specifies the logical port number.
Reserved	17.[7:1]	0x0	Reserved. Must be 0x0.
Logical Port Number Valid	17.[0]		Logical Port Number Valid 0b = Logical port number invalid. 1b = Logical port number valid.
Context	18.[7:4]		Context Software indicates the context within which the handle should be identified. 0 = Global The <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of the entire topology including motherboard and mezzanine cards. 1 = Board The <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single board. This could be motherboard or a mezzanine card. 2 = Port The <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single port, as specified in the <i>Logical Port Number</i> field. 3 = Node The <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single node (i.e. the requested node must be connected in the topology netlist using I/O connection to a specified reference node). All other values are reserved.

Table 3-161. Get Link Topology Handle Command Inputs [continued]

Argument	Byte.[Bits]	Value	Description
Node Type	18.[3:0]		Node Type Software indicates the requested node type. For the list of available node types, see Section 3.8.6.2.3.1, "Node Type and Section Length (0x0000)" .
Node Index	19.[7:0]		Node Index Software indicates the requested node index. When a PHY device is requested and the <i>Context</i> of the search is "Port", PHYs are indexed according to their place in the PHY chain where index 0 is associated with the outermost PHY. For other node types the nodes are indexed according to their appearance in the netlist. An error is returned when the index is bigger than the number of topology node retrieved in the context search.
Reserved	20:21.[15:10]	0x0	Reserved. Must be 0x0.
Reference Node Handle	20:21.[9:0]		Reference Node Handle Software uses this field to specify the reference node handle when the context of the search is "Node". For more detail, see Section 3.8.6.2.3.2, "Node Handle (0x0001)" .
Reserved	22:31.[79:0]	0x0	Reserved. Must be 0x0.

3.7.7.2.2 Get Link Topology Handle Command Outputs

Table 3-162 details the output arguments of the Get Link Topology Handle command.

Table 3-162. Get Link Topology Handle Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x06E0	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.7.2.3, "Get Link Topology Handle Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:19.[31:0]	0x0	Reserved. Must be 0x0.
Reserved	20:21.[15:10]	0x0	Reserved. Must be 0x0.
Node Handle	20:21.[9:0]		Node Handle Returns the requested node handle (a pointer to a node described in the link topology structure).
Node Part Number	22.[7:0]		Node Part Number The part number as appears in the topology netlist for this device.
Reserved	23:31.[71:0]	0x0	Reserved. Must be 0x0.

3.7.7.2.3 Get Link Topology Handle Command Errors

The following errors can be returned by the Get Link Topology Handle command:

Table 3-163. Get Link Topology Handle Command Errors

Error Code	Value	Description
OK	0	Success. No error.
ENOENT	2	No such element. - Node handle was not found (e.g. Index was higher than the number of handles found for the given context). - Uninitialized chain of PHYs for the given port.
EINVAL	14	Invalid argument. - Wrong search context. - Using "Node" context, input node handle is not valid.
ERANGE	18	Parameter out of range. Using "Port" context: - Port number is out of range. - Wrong node type (i.e. not one of PHY, Cage, LED, Temperature Sensor).

3.7.7.3 Get Link Topology Pin (0x06E1)

The Get Link Topology Pin command is used to retrieve the controlling pin/interface number for a specific pin/interface function of a topology node. The caller must provide the handle of the node that owns the I/O directly, the I/O function or Number and the I/O type. The command returns the driving node handle, the driving I/O number, and additional information regarding the pin.

3.7.7.3.1 Get Link Topology Pin Command Inputs

Table 3-164 details the input arguments of the Get Link Topology Pin command.

Table 3-164. Get Link Topology Pin Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x06E1	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Logical Port Number	16.[7:0]		Logical Port Number This field specifies the logical port number.
Reserved	17.[7:1]	0x0	Reserved. Must be 0x0.
Logical Port Number Valid	17.[0]		Logical Port Number Valid 0b = Logical port number invalid. 1b = Logical port number valid.
Context	18.[7:4]		Context Software indicates the context within which the handle should be identified. 0 = Global The <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of the entire topology including motherboard and mezzanine cards. 1 = Board The <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single board. This could be motherboard or a mezzanine card. 2 = Port The <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single port, as specified in the <i>Logical Port Number</i> field. 3 = Node The <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single node (i.e. the requested node must be connected in the topology netlist using I/O connection to a specified reference node). 4 = Provided Node Handle In this mode, the node handle is taken from the command input. All other values are reserved.
Node Type	18.[3:0]		Node Type Software indicates the requested node type. For the list of available node types, see Section 3.8.6.2.3.1, "Node Type and Section Length (0x0000)" .

Table 3-164. Get Link Topology Pin Command Inputs [continued]

Argument	Byte.[Bits]	Value	Description
Node Index	19.[7:0]		Node Index Software indicates the requested node index. When a PHY device is requested and the <i>Context</i> of the search is "Port", PHYs are indexed according to their place in the PHY chain where index 0 is associated with the outermost PHY. For other node types the nodes are indexed according to their appearance in the netlist. An error is returned when the index is bigger than the number of topology node retrieved in the context search.
Reserved	20:21.[15:10]	0x0	Reserved. Must be 0x0.
Reference Node Handle / Node Handle	20:21.[9:0]		Reference Node Handle / Node Handle When <i>Context</i> is "Node", specifies the reference node handle to be used to identify the node the pin belongs to. When the <i>Context</i> is "Provided Node Handle", represents the handle identifying the node the pin belongs to and no search is conducted The value is ignored in all other cases.
I/O Type	22.[7:5]		I/O Type The type of the I/O. For the list of available I/O types, see Section 3.8.6.2.4.2, "I/O Type and Driving Interface (0x0001 + n*2)" .
I/O Function / Number	22.[4:0]		I/O Function / Number The I/O function or number. For the list of available I/O Functions, see Section 3.8.6.2.4.2, "I/O Type and Driving Interface (0x0001 + n*2)" .
Reserved	23:31.[71:0]	0x0	Reserved. Must be 0x0.

3.7.7.3.2 Get Link Topology Pin Command Outputs

Table 3-165 details the output arguments of the Get Link Topology Pin command.

Table 3-165. Get Link Topology Pin Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x06E1	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.7.3.3, "Get Link Topology Pin Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:19.[31:0]	0x0	Reserved. Must be 0x0.
Reserved	20:21.[15:10]	0x0	Reserved. Must be 0x0.
Driving Node Handle	20:21.[9:0]		Driving Node Handle The handle of the link topology node the identified pin/interface is connected to. To identify whether this node is actually configured as an output and driving the signal or is configured as an input, refer to the returned I/O Function.
Reserved	22.[7:0]	0x0	Reserved. Must be 0x0
Reserved	23.[7:6]	00b	Reserved. Must be 00b.

Table 3-165. Get Link Topology Pin Command Outputs [continued]

Argument	Byte.[Bits]	Value	Description
Driving I/O Number	23.[5:0]		Driving I/O Number The number of the interface with the specified I/O Type on the device pointed to by Driving Node Handle the connection leads to. For more detail, see Section 3.8.6.2.4.1, "Driving Node Handle (0x0000 + n*2)" .
Driven	24.[7]		Driven Reflects whether the I/O is strapped to a certain value or is driven by another node in the topology. For more detail, see Section 3.8.6.2.4.2, "I/O Type and Driving Interface (0x0001 + n*2)" .
Value	24.[6]		Value Reflects the default or static value of the I/O. Only applicable to I/Os with I/O Type of GPIO and MDIO Address. For more detail, see Section 3.8.6.2.4.2, "I/O Type and Driving Interface (0x0001 + n*2)" .
Polarity	24.[5]		Polarity Polarity configuration for the I/O. Only applicable to I/Os with I/O Type of GPIO. For more detail, see Section 3.8.6.2.4.2, "I/O Type and Driving Interface (0x0001 + n*2)" .
Interrupt	24.[4:3]		Interrupt Reflects the interrupt configuration of the I/O. Only applicable to I/Os with I/O Type of GPIO. For more detail, see Section 3.8.6.2.4.2, "I/O Type and Driving Interface (0x0001 + n*2)" .
I2C/MDIO Speed	24.[2:0]		I2C/MDIO Speed The bus speed. Only applicable to I/Os with I/O Type of I2C Bus and MDIO Bus. For more detail, see Section 3.8.6.2.4.2, "I/O Type and Driving Interface (0x0001 + n*2)" .
Reserved	25:31.[55:0]	0x0	Reserved. Must be 0x0

3.7.7.3.3 Get Link Topology Pin Command Errors

The following errors can be returned by the Get Link Topology Pin command:

Table 3-166. Get Link Topology Pin Command Errors

Error Code	Value	Description
OK	0	Success. No error.
ENOENT	2	No such element. - Node handle was not found (e.g. Index was higher than the number of handles found for the given context). - Uninitialized chain of PHYs for the given port.
ENXIO	6	No such resource. The given I/O type was not found for the given node handle.
EFAULT	11	Bad address. For I/O Type GPIO, illegal GPIO function.
ERANGE	18	Parameter out of range. The given I/O Number is out of bounds for a given node handle and given I/O type. Using "Port" context: - Port number is out of range. - Wrong node type (i.e. not one of PHY, Cage, LED, Temperature Sensor).

3.7.7.4 Read I2C (0x06E2)

The Read I2C command is used to read from a device connected to an I²C bus, which is part of the link topology. The caller can provide the handle of the node it wants to access in which case the location and address of the node within the topology is abstracted. Alternatively the caller can provide the I²C Bus and I²C Address directly.

3.7.7.4.1 Read I2C Command Inputs

Table 3-167 details the input arguments of the Read I2C command.

Table 3-167. Read I2C Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x06E2	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Logical Port Number	16.[7:0]		Logical Port Number This field specifies the logical port number.
Reserved	17.[7:1]	0x0	Reserved. Must be 0x0.
Logical Port Number Valid	17.[0]		Logical Port Number Valid 0b = Logical port number invalid. 1b = Logical port number valid.

Table 3-167. Read I2C Command Inputs [continued]

Argument	Byte.[Bits]	Value	Description
Context	18.[7:4]		Context Indicates the context within which the node should be identified and the mode of access. 0 = Global In this mode, the <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of the entire topology including motherboard and mezzanine cards. The path to the node is established automatically as required, and the I²C Bus and I²C Address for the transaction are provided by the link topology netlist. 1 = Board In this mode, the <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single board. This could be motherboard or a mezzanine card. The path to the node is established automatically as required, and the I²C Bus and I²C Address for the transaction are provided by the link topology netlist. 2 = Port In this mode, the <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single port, as specified in the <i>Logical Port Number</i> field. 3 = Node In this mode, the <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single node (i.e. the requested node must be connected in the topology netlist using I/O connection to a specified reference node). 4 = Provided Node Handle In this mode, the node handle is taken from the command input. The path to the node is established automatically as required, and the I²C Bus and I²C Address are taken from the link topology netlist. 5 = Direct Bus Access In this mode, no topology search is done. The path to the node needs to be manually established, and the I²C Bus and I²C Address for the transaction are taken from the command input. 6 = Provided Node Handle and Bus Address In this mode, the node handle is taken from the command input. The path to the node is established automatically as required, the I²C Bus is taken from the link topology netlist, and the I²C Address is taken from the command input. All other values are reserved.
Node Type	18.[3:0]		Node Type Indicates the requested node type. For the list of available node types, see Section 3.8.6.2.3.1, "Node Type and Section Length (0x0000)".
Node Index / Bus Number	19.[7:0]		Node Index / Bus Number Indicates the requested node index. When a PHY device is requested and the context of the search is "Port", PHYs are indexed according to their place in the PHY chain where index 0 is associated with the outermost PHY. For other node types, the nodes are indexed according to their appearance in the netlist. An error is returned when the index is bigger than the number of topology node retrieved in the context search When the <i>Context</i> is "Direct Bus Access", it represents the I²C bus number (0:1).
Reserved	20:21.[15:10]	0x0	Reserved. Must be 0x0.

Table 3-167. Read I2C Command Inputs [continued]

Argument	Byte.[Bits]	Value	Description
Reference Node Handle / Node Handle	20:21.[9:0]		Reference Node Handle / Node Handle When <i>Context</i> is "Node", specifies the reference node handle to be used to identify the node the transactions are directed to. When <i>Context</i> is "Provided Node Handle" or "Provided Node Handle and Bus Address", represents the handle of the link topology node the transactions are directed to. The value is ignored in all other cases.
I2C Offset	22:23.[15:0]		I²C Offset The memory address from which to start the read.
Repeated Start	24.[7]		Repeated Start Repeated start control. 0b = Do not use repeated start, use stop start in stead. 1b = Use repeated start.
I2C Offset Size	24.[6:5]		I²C Offset Size The number of offset bytes to write (0:2 bytes).
I2C Address Size	24.[4]		I²C Address Size The I ² C bus address size in bits. 0b = 7-bit address. 1b = 10-bit address.
Data Length	24.[3:0]		Data Length The number of bytes to read (1:16 bytes). To represent the maximum allowed data length, the following mapping is used: 0b = 16 bytes 1b = 1:15 bytes
Reserved	25.[7:0]	0x0	Reserved. Must be 0x0.
Reserved	26:27.[15:10]	0x0	Reserved. Must be 0x0.
I2C Bus Address	26:27.[9:0]		I²C Bus Address Used when <i>Context</i> is "Provided Node Handle and Bus Address" or "Direct Bus Access", ignored otherwise.
Reserved	28:31.[31:0]	0x0	Reserved. Must be 0x0.

3.7.7.4.2 Read I2C Command Outputs

Table 3-168 details the output arguments of the Read I2C command.

Table 3-168. Read I2C Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x06E2	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.7.4.3, "Read I2C Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
I2C Data	16:31.[127:0]		I²C Data The data read from the specified offset. Only the lower order <i>Data Length</i> number of bytes are valid.

3.7.7.4.3 Read I2C Command Errors

The following errors can be returned by the Read I2C command:

Table 3-169. Read I2C Command Errors

Error Code	Value	Description
OK	0	Success. No error.
EBUSY	12	Device or resource busy. The I ² C device is non-responsive or is being accessed by the Link Manager.
EINVAL	14	Invalid argument.

3.7.7.5 Write I2C (0x06E3)

The Write I2C command is used to write to a device connected to an I²C bus, which is part of the link topology. The caller can provide the handle of the node it wants to access in which case the location and address of the node within the topology is abstracted. Alternatively the caller can provide the I²C Bus and I²C Address directly.

3.7.7.5.1 Write I2C Command Inputs

Table 3-170 details the input arguments of the Write I2C command.

Table 3-170. Write I2C Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x06E3	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Logical Port Number	16.[7:0]		Logical Port Number This field specifies the logical port number.
Reserved	17.[7:1]	0x0	Reserved. Must be 0x0.
Logical Port Number Valid	17.[0]		Logical Port Number Valid 0b = Logical port number invalid. 1b = Logical port number valid.

Table 3-170. Write I2C Command Inputs [continued]

Argument	Byte.[Bits]	Value	Description
Context	18.[7:4]		Context Indicates the context within which the node should be identified and the mode of access. 0 = Global In this mode, the <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of the entire topology including motherboard and mezzanine cards. The path to the node is established automatically as required, and the I²C Bus and I²C Address for the transaction are provided by the link topology netlist. 1 = Board In this mode, the <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single board. This could be motherboard or a mezzanine card. The path to the node is established automatically as required, and the I²C Bus and I²C Address for the transaction are provided by the link topology netlist. 2 = Port In this mode, the <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single port, as specified in the <i>Logical Port Number</i> field. 3 = Node In this mode, the <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single node (i.e. the requested node must be connected in the topology netlist using I/O connection to a specified reference node). 4 = Provided Node Handle In this mode, the node handle is taken from the command input. The path to the node is established automatically as required, and the I²C Bus and I²C Address are taken from the link topology netlist. 5 = Direct Bus Access In this mode, no topology search is done. The path to the node needs to be manually established, and the I²C Bus and I²C Address for the transaction are taken from the command input. 6 = Provided Node Handle and Bus Address In this mode, the node handle is taken from the command input. The path to the node is established automatically as required, the I²C Bus is taken from the link topology netlist, and the I²C Address is taken from the command input. All other values are reserved.
Node Type	18.[3:0]		Node Type Indicates the requested node type. For the list of available node types, see Section 3.8.6.2.3.1, "Node Type and Section Length (0x0000)".
Node Index / Bus Number	19.[7:0]		Node Index / Bus Number Indicates the requested node index. When a PHY device is requested and the context of the search is "Port", PHYs are indexed according to their place in the PHY chain where index 0 is associated with the outermost PHY. For other node types, the nodes are indexed according to their appearance in the netlist. An error is returned when the index is bigger than the number of topology node retrieved in the context search When the <i>Context</i> is "Direct Bus Access", it represents the I²C bus number (0:1).
Reserved	20:21.[15:10]	0x0	Reserved. Must be 0x0.

Table 3-170. Write I2C Command Inputs [continued]

Argument	Byte.[Bits]	Value	Description
Reference Node Handle / Node Handle	20:21.[9:0]		Reference Node Handle / Node Handle When <i>Context</i> is "Node", specifies the reference node handle to be used to identify the node the transactions are directed to. When <i>Context</i> is "Provided Node Handle" or "Provided Node Handle and Bus Address", represents the handle of the link topology node the transactions are directed to. The value is ignored in all other cases.
I2C Offset	22:23.[15:0]		I²C Offset The memory address to write to.
Reserved	24.[7]	0b	Reserved. Must be 0b.
I2C Offset Size	24.[6:5]		I²C Offset Size The number of offset bytes to write (0:2 bytes).
I2C Address Size	24.[4]		I²C Address Size The I ² C bus address size in bits. 0b = 7-bit address. 1b = 10-bit address.
Data Length	24.[3:0]		Data Length The number of bytes to write (1:4 bytes).
Reserved	25.[7:0]	0x0	Reserved. Must be 0x0.
Reserved	26:27.[15:10]	0x0	Reserved. Must be 0x0.
I2C Bus Address	26:27.[9:0]		I²C Bus Address Used when <i>Context</i> is "Provided Node Handle and Bus Address" or "Direct Bus Access", ignored otherwise.
I2C Data	28:31.[31:0]		I²C Data The data to write to the specified offset. Only the lower order <i>Data Length</i> number of bytes are written.

3.7.7.5.2 Write I2C Command Outputs

Table 3-171 details the output arguments of the Write I2C command.

Table 3-171. Write I2C Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x06E3	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.7.5.3, "Write I2C Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:31.[127:0]	0x0	Reserved. Must be 0x0.

3.7.7.5.3 Write I2C Command Errors

The following errors can be returned by the Write I2C command:

Table 3-172. Write I2C Command Errors

Error Code	Value	Description
OK	0	Success. No error.
EBUSY	12	Device or resource busy. The I ² C device is non-responsive or is being accessed by the Link Manager.
EINVAL	14	Invalid argument.

3.7.7.6 Read MDIO (0x06E4)

The Read MDIO command is used to read from a device connected to an MDIO bus, which is part of the link topology. The caller can provide the handle of the node it wants to access in which case the location and address of the node within the topology is abstracted. Alternatively the caller can provide the MDIO Bus and MDIO Bus Address directly.

3.7.7.6.1 Read MDIO Command Inputs

Table 3-173 details the input arguments of the Read MDIO command.

Table 3-173. Read MDIO Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x06E4	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Logical Port Number	16.[7:0]		Logical Port Number This field specifies the logical port number.
Reserved	17.[7:1]	0x0	Reserved. Must be 0x0.
Logical Port Number Valid	17.[0]		Logical Port Number Valid 0b = Logical port number invalid. 1b = Logical port number valid.

Table 3-173. Read MDIO Command Inputs [continued]

Argument	Byte.[Bits]	Value	Description
Context	18.[7:4]		Context Indicates the context within which the node should be identified and the mode of access. 0 = Global In this mode, the <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of the entire topology including motherboard and mezzanine cards. The path to the node is established automatically as required, and the I²C Bus and I²C Address for the transaction are provided by the link topology netlist. 1 = Board In this mode, the <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single board. This could be motherboard or a mezzanine card. The path to the node is established automatically as required, and the I²C Bus and I²C Address for the transaction are provided by the link topology netlist. 2 = Port In this mode, the <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single port, as specified in the <i>Logical Port Number</i> field. 3 = Node In this mode, the <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single node (i.e. the requested node must be connected in the topology netlist using I/O connection to a specified reference node). 4 = Provided Node Handle In this mode, the node handle is taken from the command input. The path to the node is established automatically as required, and the I²C Bus and I²C Address are taken from the link topology netlist. 5 = Direct Bus Access In this mode, no topology search is done. The path to the node needs to be manually established, and the I²C Bus and I²C Address for the transaction are taken from the command input. 6 = Provided Node Handle and Bus Address In this mode, the node handle is taken from the command input. The path to the node is established automatically as required, the I²C Bus is taken from the link topology netlist, and the I²C Address is taken from the command input. All other values are reserved.
Node Type	18.[3:0]		Node Type Indicates the requested node type. For the list of available node types, see Section 3.8.6.2.3.1, "Node Type and Section Length (0x0000)".
Node Index / Bus Number	19.[7:0]		Node Index / Bus Number The MDIO bus number (0:1). Used when <i>Context</i> is "Direct Bus Access", ignored otherwise.
Reserved	20:21.[15:10]	0x0	Reserved. Must be 0x0.
Reference Node Handle / Node Handle	20:21.[9:0]		Reference Node Handle / Node Handle When <i>Context</i> is "Node", specifies the reference node handle to be used to identify the node the transactions are directed to. When <i>Context</i> is "Provided Node Handle" or "Provided Node Handle and Bus Address", represents the handle of the link topology node the transactions are directed to. The value is ignored in all other cases.
Reserved	22.[7]	0b	Reserved. Must be 0b.

Table 3-173. Read MDIO Command Inputs [continued]

Argument	Byte.[Bits]	Value	Description
MDIO Protocol	22.[6:5]		MDIO Protocol The MDIO protocol to be used. 01b = Clause 22 MDIO formatting. 10b = Clause 45 MDIO formatting. All other values are reserved.
MDIO Device Address	22.[4:0]		MDIO Device Address The address of the MDIO device from which the data should be read.
Reserved	23.[7:5]	000b	Reserved. Must be 000b.
MDIO BUS Address	23.[4:0]		MDIO BUS Address Used when <i>Context</i> is "Provided Node Handle and Bus Address" or "Direct Bus Access", ignored otherwise
Offset	24:25.[15:0]		Offset The memory address from which to start the read.
Reserved	26:31.[127:0]	0x0	Reserved. Must be 0x0.

3.7.7.6.2 Read MDIO Command Outputs

Table 3-174 details the output arguments of the Read MDIO command.

Table 3-174. Read MDIO Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x06E4	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.7.6.3, "Read MDIO Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:17.[15:0]	0x0	Reserved. Must be 0x0.

Table 3-174. Read MDIO Command Outputs [continued]

Argument	Byte.[Bits]	Value	Description
Context	18.[7:4]		Context Indicates the context within which the node should be identified and the mode of access. 0 = Global In this mode, the <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of the entire topology including motherboard and mezzanine cards. The path to the node is established automatically as required, and the I²C Bus and I²C Address for the transaction are provided by the link topology netlist. 1 = Board In this mode, the <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single board. This could be motherboard or a mezzanine card. The path to the node is established automatically as required, and the I²C Bus and I²C Address for the transaction are provided by the link topology netlist. 2 = Port In this mode, the <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single port, as specified in the <i>Logical Port Number</i> field. 3 = Node In this mode, the <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single node (i.e. the requested node must be connected in the topology netlist using I/O connection to a specified reference node). 4 = Provided Node Handle In this mode, the node handle is taken from the command input. The path to the node is established automatically as required, and the I²C Bus and I²C Address are taken from the link topology netlist. 5 = Direct Bus Access In this mode, no topology search is done. The path to the node needs to be manually established, and the I²C Bus and I²C Address for the transaction are taken from the command input. 6 = Provided Node Handle and Bus Address In this mode, the node handle is taken from the command input. The path to the node is established automatically as required, the I²C Bus is taken from the link topology netlist, and the I²C Address is taken from the command input. All other values are reserved.
Node Type	18.[3:0]		Node Type Indicates the requested node type. For the list of available node types, see Section 3.8.6.2.3.1, "Node Type and Section Length (0x0000)".
Node Index / Bus Number	19.[7:0]		Node Index / Bus Number The MDIO bus number (0:1). Used when <i>Context</i> is "Direct Bus Access", ignored otherwise.
Reserved	20:21.[15:10]	0x0	Reserved. Must be 0x0.
Reference Node Handle / Node Handle	20:21.[9:0]		Reference Node Handle / Node Handle When <i>Context</i> is "Node", specifies the reference node handle to be used to identify the node the transactions are directed to. When <i>Context</i> is "Provided Node Handle" or "Provided Node Handle and Bus Address", represents the handle of the link topology node the transactions are directed to. The value is ignored in all other cases.
Reserved	22.[7]	0b	Reserved. Must be 0b.

Table 3-174. Read MDIO Command Outputs [continued]

Argument	Byte.[Bits]	Value	Description
MDIO Protocol	22.[6:5]		MDIO Protocol The MDIO protocol to be used. 01b = Clause 22 MDIO formatting. 10b = Clause 45 MDIO formatting. All other values are reserved.
MDIO Device Address	22.[4:0]		MDIO Device Address The address of the MDIO device from which the data should be read.
Reserved	23.[7:5]	000b	Reserved. Must be 000b.
MDIO BUS Address	23.[4:0]		MDIO BUS Address Used when <i>Context</i> is "Provided Node Handle and Bus Address" or "Direct Bus Access", ignored otherwise
Offset	24:25.[15:0]		Offset The memory address from which to start the read.
Data	26:27.[15:0]		Data The data read from the specified offset.
Reserved	28:31.[31:0]	0x0	Reserved. Must be 0x0.

3.7.7.6.3 Read MDIO Command Errors

The following errors can be returned by the Read MDIO command:

Table 3-175. Read MDIO Command Errors

Error Code	Value	Description
OK	0	Success. No error.
EBUSY	12	Device or resource busy. The MDIO device is non-responsive or is being accessed by the Link Manager.
EINVAL	14	Invalid argument.

3.7.7.7 Write MDIO (0x06E5)

The Write MDIO command is used to write to a device connected to an MDIO bus, which is part of the link topology. The caller can provide the handle of the node it wants to access in which case the location and address of the node within the topology is abstracted. Alternatively the caller can provide the MDIO Bus and MDIO Bus Address directly.

3.7.7.7.1 Write MDIO Command Inputs

Table 3-176 details the input arguments of the Write MDIO command.

Table 3-176. Write MDIO Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x06E5	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Logical Port Number	16.[7:0]		Logical Port Number This field specifies the logical port number.
Reserved	17.[7:1]	0x0	Reserved. Must be 0x0.
Logical Port Number Valid	17.[0]		Logical Port Number Valid 0b = Logical port number invalid. 1b = Logical port number valid.

Table 3-176. Write MDIO Command Inputs [continued]

Argument	Byte.[Bits]	Value	Description
Context	18.[7:4]		Context Indicates the context within which the node should be identified and the mode of access. 0 = Global In this mode, the <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of the entire topology including motherboard and mezzanine cards. The path to the node is established automatically as required, and the I²C Bus and I²C Address for the transaction are provided by the link topology netlist. 1 = Board In this mode, the <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single board. This could be motherboard or a mezzanine card. The path to the node is established automatically as required, and the I²C Bus and I²C Address for the transaction are provided by the link topology netlist. 2 = Port In this mode, the <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single port, as specified in the <i>Logical Port Number</i> field. 3 = Node In this mode, the <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single node (i.e. the requested node must be connected in the topology netlist using I/O connection to a specified reference node). 4 = Provided Node Handle In this mode, the node handle is taken from the command input. The path to the node is established automatically as required, and the I²C Bus and I²C Address are taken from the link topology netlist. 5 = Direct Bus Access In this mode, no topology search is done. The path to the node needs to be manually established, and the I²C Bus and I²C Address for the transaction are taken from the command input. 6 = Provided Node Handle and Bus Address In this mode, the node handle is taken from the command input. The path to the node is established automatically as required, the I²C Bus is taken from the link topology netlist, and the I²C Address is taken from the command input. All other values are reserved.
Node Type	18.[3:0]		Node Type Indicates the requested node type. For the list of available node types, see Section 3.8.6.2.3.1, "Node Type and Section Length (0x0000)".
Node Index / Bus Number	19.[7:0]		Node Index / Bus Number The MDIO bus number (0:1). Used when <i>Context</i> is "Direct Bus Access", ignored otherwise.
Reserved	20:21.[15:10]	0x0	Reserved. Must be 0x0.
Reference Node Handle / Node Handle	20:21.[9:0]		Reference Node Handle / Node Handle When <i>Context</i> is "Node", specifies the reference node handle to be used to identify the node the transactions are directed to. When <i>Context</i> is "Provided Node Handle" or "Provided Node Handle and Bus Address", represents the handle of the link topology node the transactions are directed to. The value is ignored in all other cases.
Reserved	22.[7]	0b	Reserved. Must be 0b.

Table 3-176. Write MDIO Command Inputs [continued]

Argument	Byte.[Bits]	Value	Description
MDIO Protocol	22.[6:5]		MDIO Protocol The MDIO protocol to be used. 01b = Clause 22 MDIO formatting. 10b = Clause 45 MDIO formatting. All other values are reserved.
MDIO Device Address	22.[4:0]		MDIO Device Address The address of the MDIO device from which the data should be read.
Reserved	23.[7:5]	000b	Reserved. Must be 000b.
MDIO BUS Address	23.[4:0]		MDIO BUS Address Used when <i>Context</i> is "Provided Node Handle and Bus Address" or "Direct Bus Access", ignored otherwise
Offset	24:25.[15:0]		Offset The memory address from which to start the read.
Data	26:27.[15:0]		Data The data to be written to the specified offset.
Reserved	28:31.[31:0]	0x0	Reserved. Must be 0x0.

3.7.7.7.2 Write MDIO Command Outputs

Table 3-177 details the output arguments of the Write MDIO command.

Table 3-177. Write MDIO Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x06E5	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.7.7.3, "Write MDIO Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:31.[127:0]	0x0	Reserved. Must be 0x0.

3.7.7.7.3 Write MDIO Command Errors

The following errors can be returned by the Write MDIO command:

Table 3-178. Write MDIO Command Errors

Error Code	Value	Description
OK	0	Success. No error.
EBUSY	12	Device or resource busy. The MDIO device is non-responsive or is being accessed by the Link Manager.
EINVAL	14	Invalid argument.

3.7.7.8 Set GPIO By Function (0x06E6)

The Set GPIO By Function command is used to set a GPIO signal which is part of the link topology. The GPIO signal is referenced by its function within the netlist (i.e. it can be used to set the RESET_N signal of specific topology node). The caller needs to provide the handle of the node that owns the GPIO directly and the I/O Function and the value to set.

3.7.7.8.1 Set GPIO By Function Command Inputs

Table 3-179 details the input arguments of the Set GPIO By Function command.

Table 3-179. Set GPIO By Function Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x06E6	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Logical Port Number	16.[7:0]		Logical Port Number This field specifies the logical port number.
Reserved	17.[7:1]	0x0	Reserved. Must be 0x0.
Logical Port Number Valid	17.[0]		Logical Port Number Valid 0b = Logical port number invalid. 1b = Logical port number valid.
Context	18.[7:4]		Context Indicates the context within which the handle should be identified. 0 = Global The <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of the entire topology including motherboard and mezzanine cards. 1 = Board The <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single board. This could be motherboard or a mezzanine card. 2 = Port The <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single port, as specified in the <i>Logical Port Number</i> field. 3 = Node The <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single node (i.e. the requested node must be connected in the topology netlist using I/O connection to a specified reference node). 4 = Provided Node The node handle is provided as an input and no search is conducted. All other values are reserved.
Node Type	18.[3:0]		Node Type Indicates the requested node type. For the list of available node types, see Section 3.8.6.2.3.1, "Node Type and Section Length (0x0000)" .

Table 3-179. Set GPIO By Function Command Inputs [continued]

Argument	Byte.[Bits]	Value	Description
Node Index	19.[7:0]		Node Index Indicates the requested node index. When a PHY device is requested and the context of the search is "Port", PHYs are indexed according to their place in the PHY chain where index 0 is associated with the outermost PHY. For other node types the nodes are indexed according to their appearance in the netlist. An error is returned when the index is bigger than the number of topology node retrieved in the context search.
Reserved	20:21.[15:10]	0x0	Reserved. Must be 0x0.
Reference Node Handle / Node Handle	20:21.[9:0]		Reference Node Handle / Node Handle When <i>Context</i> is "Node" specifies the reference node handle to be used to identify the node the pin belongs to. When the <i>Context</i> is "Provided Node Handle" represents the handle identifying the node the pin belongs to and no search is conducted. The value is ignored in all other cases.
Reserved	22.[7:5]	000b	Reserved. Must be 000b.
I/O Function	22.[4:0]		I/O Function The I/O function of the GPIO to configure on the specified node. For the list of available I/O functions, see Section 3.8.6.2.4.2, "I/O Type and Driving Interface (0x0001 + n*2)" .
Reserved	23.[7:1]	0x0	Reserved. Must be 0x0.
I/O Value	23.[0]		I/O Value The logical I/O value to configure. 0b = De-asserted 1b = Asserted Note: The GPIO polarity is taken from the link topology netlist.
Reserved	24:31.[63:0]	0x0	Reserved. Must be 0x0.

3.7.7.8.2 Set GPIO By Function Command Outputs

Table 3-177 details the output arguments of the Set GPIO By Function command.

Table 3-180. Set GPIO By Function Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x06E6	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.7.8.3, "Set GPIO By Function Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:31.[127:0]	0x0	Reserved. Must be 0x0.

3.7.7.8.3 Set GPIO By Function Command Errors

The following errors can be returned by the Set GPIO By Function command:

Table 3-181. Set GPIO By Function Command Errors

Error Code	Value	Description
OK	0	Success. No error.
ENOENT	2	Not found. Node or GPIO not found.
EBUSY	12	Device or resource busy. The device is non-responsive or is being accessed by the Link Manager.
EINVAL	14	Invalid argument. Invalid I/O Function.
EMODE	21	GPIO is input.

3.7.7.9 Get GPIO By Function (0x06E7)

The Get GPIO By Function command is used to read the value of a GPIO signal which is part of the link topology. The GPIO signal is referenced by its function within the netlist (i.e. it can be used to set the INT_N signal of specific topology node). The caller needs to provide the handle of the node that owns the GPIO directly and the I/O Function.

3.7.7.9.1 Get GPIO By Function Command Inputs

Table 3-182 details the input arguments of the Get GPIO By Function command.

Table 3-182. Get GPIO By Function Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x06E7	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Logical Port Number	16.[7:0]		Logical Port Number This field specifies the logical port number.
Reserved	17.[7:1]	0x0	Reserved. Must be 0x0.
Logical Port Number Valid	17.[0]		Logical Port Number Valid 0b = Logical port number invalid. 1b = Logical port number valid.
Context	18.[7:4]		Context Software indicates the context within which the handle should be identified. 0 = Global The <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of the entire topology including motherboard and mezzanine cards. 1 = Board The <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single board. This could be motherboard or a mezzanine card. 2 = Port The <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single port, as specified in the <i>Logical Port Number</i> field. 3 = Node The <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single node (i.e. the requested node must be connected in the topology netlist using I/O connection to a specified reference node). 4 = Provided Node The node handle is provided as an input and no search is conducted. All other values are reserved.
Node Type	18.[3:0]		Node Type Indicates the requested node type. For the list of available node types, see Section 3.8.6.2.3.1, "Node Type and Section Length (0x0000)" .

Table 3-182. Get GPIO By Function Command Inputs [continued]

Argument	Byte.[Bits]	Value	Description
Node Index	19.[7:0]		Node Index Indicates the requested node index. When a PHY device is requested and the context of the search is "Port", PHYs are indexed according to their place in the PHY chain where index 0 is associated with the outermost PHY. For other node types the nodes are indexed according to their appearance in the netlist. An error is returned when the index is bigger than the number of topology node retrieved in the context search.
Reserved	20:21.[15:10]	0x0	Reserved. Must be 0x0.
Reference Node Handle / Node Handle	20:21.[9:0]		Reference Node Handle / Node Handle When <i>Context</i> is "Node", specifies the reference node handle to be used to identify the node the pin belongs to. When the <i>Context</i> is "Provided Node Handle", represents the handle identifying the node the pin belongs to and no search is conducted. The value is ignored in all other cases.
Reserved	22.[7:5]	000b	Reserved. Must be 000b.
I/O Function	22.[4:0]		I/O Function The I/O function of the GPIO to read on the specified node. For the list of available I/O functions, see Section 3.8.6.2.4.2, "I/O Type and Driving Interface (0x0001 + n*2)" .
Reserved	23:31.[71:0]	0x0	Reserved. Must be 0x0.

3.7.7.9.2 Get GPIO By Function Command Outputs

Table 3-183 details the output arguments of the Get GPIO By Function command.

Table 3-183. Get GPIO By Function Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x06E7	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.7.9.3, "Get GPIO By Function Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:22.[55:0]	0x0	Reserved. Must be 0x0.
Reserved	23.[7:1]	0x0	Reserved. Must be 0x0.
I/O Value	23.[0]		I/O Value The current logical I/O value. 0b = De-asserted 1b = Asserted Note: The GPIO polarity is taken from the link topology netlist.
Reserved	24:31.[63:0]	0x0	Reserved. Must be 0x0.

3.7.7.9.3 Get GPIO By Function Command Errors

The following errors can be returned by the Get GPIO By Function command:

Table 3-184. Get GPIO By Function Command Errors

Error Code	Value	Description
OK	0	Success. No error.
EBUSY	12	Device or resource busy. The device is non-responsive or is being accessed by the Link Manager.
EINVAL	14	Invalid argument. Invalid <i>I/O Function</i> .

3.7.7.10 Set LED (0x06E8)

The Set LED command is used to set an LED signal which is part of the link topology. The caller must provide the node handle of the LED node directly, the LED value to set and the blink option.

Note: When the LED is defined at the topology netlist and is associated with a port, the Link Manager might override the value set by this command. To use an LED for arbitrary purposes, the LED must not be associated with any ports defined within the link topology.

3.7.7.10.1 Set LED Command Inputs

Table 3-185 details the input arguments of the Set LED command.

Table 3-185. Set LED Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x06E8	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Logical Port Number	16.[7:0]		Logical Port Number This field specifies the logical port number.
Reserved	17.[7:1]	0x0	Reserved. Must be 0x0.
Logical Port Number Valid	17.[0]		Logical Port Number Valid 0b = Logical port number invalid. 1b = Logical port number valid.
Context	18.[7:4]		Context Software indicates the context within which the handle should be identified. 0 = Global The <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of the entire topology including motherboard and mezzanine cards. 1 = Board The <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single board. This could be motherboard or a mezzanine card. 2 = Port The <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single port, as specified in the <i>Logical Port Number</i> field. 3 = Node The <i>Node Index</i> and <i>Node Type</i> are used to identify the node within the context of a single node (i.e. the requested node must be connected in the topology netlist using I/O connection to a specified reference node). 4 = Provided Node The node handle is provided as an input and no search is conducted. All other values are reserved.
Node Type	18.[3:0]		Node Type Indicates the requested node type. For the list of available node types, see Section 3.8.6.2.3.1, "Node Type and Section Length (0x0000)" .

Table 3-185. Set LED Command Inputs [continued]

Argument	Byte.[Bits]	Value	Description
Node Index	19.[7:0]		Node Index Indicates the requested LED node index. LEDs are indexed according to their appearance in the netlist. An error is returned when the index is bigger than the number of topology node retrieved in the context search.
Reserved	20:21.[15:10]	0x0	Reserved. Must be 0x0.
Reference Node Handle / Node Handle	20:21.[9:0]		Reference Node Handle / Node Handle When the <i>Context</i> is "Node", specifies the reference node handle to be used to identify the LED node. When the <i>Context</i> is "Provided Node Handle", represents the handle identifying the LED node The value is ignored in all other cases.
Reserved	22.[7:6]	00b	Reserved. Must be 00b.
LED Blink	22.[5:3]		LED Blink The LED blink configuration. 0 = No blink 1 = Blink Slow 5 = Blink Fast All other values are reserved. Slow blinking results in a cadence of 200 ms on / 200 ms off. Fast blinking results in a cadence of 100 ms on / 100 ms off.
LED Value	22.[2:0]		LED Value The LED color configuration. For RGB LED Nodes, the allowed values are as follows: 0 = OFF 1 = RED 2 = ORANGE 3 = YELLOW 4 = GREEN 5 = BLUE 6 = PURPLE All other values are reserved. Note: Colors are only supported if the <i>Node Part Number</i> is RGB LED. The node has three I/Os defined and they all are driven by the same PCA9585 LED controller. The selected color is mixed though driving the associated Red, Green and Blue LEDs with the appropriate PWM signals through a PCA9585 LED controller. For monochrome LED Nodes, the allowed values are as follows: 0 = OFF 7 = ON All other values are reserved.
Reserved	23:31.[71:0]	0x0	Reserved. Must be 0x0.

3.7.7.10.2 Set LED Command Outputs

Table 3-186 details the output arguments of the Set LED command.

Table 3-186. Set LED Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x06E8	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.7.10.3, "Set LED Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:31.[127:0]	0x0	Reserved. Must be 0x0.

3.7.7.10.3 Set LED Command Errors

The following errors can be returned by the Set LED command:

Table 3-187. Set LED Command Errors

Error Code	Value	Description
OK	0	Success. No error.
EBUSY	12	Device or resource busy. The device is non-responsive or is being accessed by the Link Manager.
EINVAL	14	Invalid argument. Invalid <i>LED Value</i> , <i>LED Blink</i> .
ERANGE	18	LED color out of range.

3.7.7.11 Set Port Identification LED (0x06E9)

The Set Port Identification LED command is used to set the LED that is used to identify the port as indicated by the link topology netlist. The caller must provide the logical port number.

3.7.7.11.1 Set Port Identification LED Command Inputs

Table 3-188 details the input arguments of the Set Port Identification LED command.

Table 3-188. Set Port Identification LED Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x06E9	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Logical Port Number	16.[7:0]		Logical Port Number This field specifies the logical port number.
Reserved	17.[7:1]	0x0	Reserved. Must be 0x0.
Logical Port Number Valid	17.[0]		Logical Port Number Valid 0b = Logical port number invalid. 1b = Logical port number valid.
Reserved	18.[7:1]	0x0	Reserved. Must be 0x0.
LED Value	18.[0]		LED Value Set the LED into identification mode. 0b = The LED is configured to its original mode (by the netlist). 1b = The LED is configured to identification mode and will blink on a 1 second cadence.
Reserved	19:31.[103:0]	0x0	Reserved. Must be 0x0.

3.7.7.11.2 Set Port Identification LED Command Outputs

Table 3-189 details the output arguments of the Set Port Identification LED command.

Table 3-189. Set Port Identification LED Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x06E9	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.7.11.3, "Set Port Identification LED Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:31.[127:0]	0x0	Reserved. Must be 0x0.

3.7.7.11.3 Set Port Identification LED Command Errors

The following errors can be returned by the Set Port Identification LED command:

Table 3-190. Set Port Identification LED Command Errors

Error Code	Value	Description
OK	0	Success. No error.
EFAULT	11	Failed to configure the LED.
EBUSY	12	Device or resource busy. The device is non-responsive or is being accessed by the Link Manager.
EINVAL	14	Invalid argument. Invalid <i>LED Value</i> , <i>LED Blink</i> .

3.7.7.12 Set GPIO (0x06EC)

The Set GPIO command is used to set a GPIO signal which is part of the link topology. The GPIO signal is referenced by a GPIO controller node within the netlist. The caller must provide the node handle of the GPIO controller, the GPIO number on the controller, and the value to set.

3.7.7.12.1 Set GPIO Command Inputs

Table 3-191 details the input arguments of the Set GPIO command.

Table 3-191. Set GPIO Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x06EC	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:17.[15:10]	0x0	Reserved. Must be 0x0.
Node Handle	16:17.[9:0]		Node Handle The node handle (a pointer to a node described in the link topology structure) identifying the GPIO controller node.
Reserved	18.[7:5]	000b	Reserved. Must be 000b.
I/O Number	18.[4:0]		I/O Number The I/O Number in the context of the specified GPIO controller node.
Reserved	19.[7:1]	0x0	Reserved. Must be 0x0.
I/O Value	19.[0]		I/O Value The I/O value to configure. 0b = Low 1b = High Note: The GPIO is treated active high.
Reserved	20:31.[95:0]	0x0	Reserved. Must be 0x0.

3.7.7.12.2 Set GPIO Command Outputs

Table 3-192 details the output arguments of the Set GPIO command.

Table 3-192. Set GPIO Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x06EC	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.7.12.3, "Set GPIO Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.

Table 3-192. Set GPIO Command Outputs [continued]

Argument	Byte.[Bits]	Value	Description
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:31.[127:0]	0x0	Reserved. Must be 0x0.

3.7.7.12.3 Set GPIO Command Errors

The following errors can be returned by the Set GPIO command:

Table 3-193. Set GPIO Command Errors

Error Code	Value	Description
OK	0	Success. No error.
EBUSY	12	Device or resource busy. The device is non-responsive or is being accessed by the Link Manager.
EINVAL	14	Invalid argument. Invalid I/O Function.

3.7.7.13 Get GPIO (0x06ED)

The Get GPIO command is used to read the value of a GPIO signal which is part of the link topology. The GPIO signal is referenced by a GPIO controller node within the netlist. The must provide the node handle of the GPIO controller and the GPIO number on the controller.

3.7.7.13.1 Get GPIO Command Inputs

Table 3-194 details the input arguments of the Get GPIO command.

Table 3-194. Get GPIO Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x06ED	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:17.[15:10]	0x0	Reserved. Must be 0x0.
Node Handle	16:17.[9:0]		Node Handle The node handle (a pointer to a node described in the link topology structure) identifying the GPIO controller node.
Reserved	18.[7:5]	000b	Reserved. Must be 000b.
I/O Number	18.[4:0]		I/O Number The I/O Number in the context of the specified GPIO controller node.
Reserved	19:31.[103:0]	0x0	Reserved. Must be 0x0.

3.7.7.13.2 Get GPIO Command Outputs

Table 3-195 details the output arguments of the Get GPIO command.

Table 3-195. Get GPIO Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x06ED	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.7.13.3, "Get GPIO Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:18.[23:0]	0x0	Reserved. Must be 0x0.
Reserved	19.[7:1]	0x0	Reserved. Must be 0x0.

Table 3-195. Get GPIO Command Outputs [continued]

Argument	Byte.[Bits]	Value	Description
I/O Value	19.[0]		I/O Value The current I/O value. 0b = Low 1b = High Note: The GPIO is treated active high.
Reserved	20:31.[95:0]	0x0	Reserved. Must be 0x0.

3.7.7.13.3 Get GPIO Command Errors

The following errors can be returned by the Get GPIO command:

Table 3-196. Get GPIO Command Errors

Error Code	Value	Description
OK	0	Success. No error.
EBUSY	12	Device or resource busy. The device is non-responsive or is being accessed by the Link Manager.
EINVAL	14	Invalid argument. Invalid I/O Function.

3.7.7.14 Read/Write SFF/CMIS Module (0x06EE)

The Read SFF/CMIS Module command is used to read or write up to 16 bytes from or to an SFF/CMIS defined pluggable module connected to an I²C bus, which is part of the link topology.

Using this command, the software driver can set the *Page* bit and the firmware checks the appropriate Page byte(s) from the EEPROM to verify operation on the right page of the module. When firmware detects that the module is currently configured to different page access, it updates the module page before the desired read or write action. The checking and setting of the module page and the actual read or write operation are done as an atomic block, where no other I²C access to this module is performed between those actions. The page aware SFF module access should be used by software to access SFF EEPROM module, as it is done in coordination with DNL preventing potential problem where the page setting and the actual read or write are interleaved between DNL and other host access.

3.7.7.14.1 Read/Write SFF/CMIS Module Command Inputs

Table 3-197 details the input arguments of the Read/Write SFF/CMIS Module command.

Table 3-197. Read/Write SFF/CMIS Module Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.2.1 for details.
Opcode	2:3.[15:0]	0x06EE	Command opcode.
Datalen	4:5.[15:0]	1-16	The number of bytes to read.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Logical Port Number	16.[7:0]		Logical Port Number This field specifies the logical port number.
Reserved	17.[7:1]	0x0	Reserved. Must be 0x0.
Logical Port Number Valid	17.[0]		Logical Port Number Valid 0b = Logical port number invalid. 1b = Logical port number valid.
Read/Write	18:19.[15]		Read/Write command 0b = Read 1b = Write
Reserved	18:19.[14:13]	00b	Reserved. Must be 00b.
Module Page/Bank Control	18:19.[12:11]		Module Page/Bank Control Configures SFF/CMIS paging and banking control. 00b = Do not update Page and Bank 01b = Update Page Link Manager reads the value of the current page (offset 127), and on mismatch updates it with the value of the configured Module Page. 10b = Update Bank Link Manager reads the value of the current bank (offset 126), and on mismatch updates it with the value of the configured Module Bank. 11b = Update Bank and Page Link Manager reads the value of the current bank and page (offsets 126, 127), and on mismatch updates them with the values of the configured Module Bank and Module Page.

Table 3-197. Read/Write SFF/CMIS Module Command Inputs [continued]

Argument	Byte.[Bits]	Value	Description
I2C Address Size	18:19.[10]		I²C Address Size The I ² C bus address size in bits. 0b = 7-bit address. 1b = 10-bit address.
I2C Bus Address	18:19.[9:0]		I²C Bus Address When this value is zero, the I ² C bus address is taken from the topology netlist.
I2C Offset	20:21.[15:0]		I²C Offset The module memory address to read from or write to.
Module Bank	22.[7:0]		Module Bank The CMIS defined bank number.
Module Page	23.[7:0]		Module Page The SFF/CMIS defined page number.
Data Address High	24:27.[31:0]		High bits of command buffer address.
Data Address Low	28:31.[31:0]		Low bits of command buffer address.

Table 3-198. Read/Write SFF/CMIS Module Command Input Buffer

Argument	Byte.[Bits]	Value	Description
I2C Data	0:15.[127:0]		I²C Data The data to be written to the specified offset. Only the lower order Data Length number of bytes are valid.

3.7.7.14.2 Read/Write SFF/CMIS Module Command Outputs

Table 3-199 details the output arguments of the Read/Write SFF/CMIS Module command.

Table 3-199. Read/Write SFF/CMIS Module Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.2.2 for details.
Opcode	2:3.[15:0]	0x06EE	Command opcode.
Datalen	4:5.[15:0]		The number of bytes read.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.7.14.3, "Read/Write SFF/CMIS Module Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:23.[63:0]	0x0	Reserved. Must be 0x0.
Data Address High	24:27.[31:0]		High bits of command buffer address.
Data Address Low	28:31.[31:0]		Low bits of command buffer address.

Table 3-200. Read/Write SFF/CMIS Module Command Output Buffer

Argument	Byte.[Bits]	Description
I2C Data	0:15.[127:0]	I²C Data The data read from the specified offset. Only the lower order Data Length number of bytes are valid.

3.7.7.14.3 Read/Write SFF/CMIS Module Command Errors

The following errors can be returned by the Read/Write SFF/CMIS Module command:

Table 3-201. Read/Write SFF/CMIS Module Command Errors

Error Code	Value	Description
OK	0	Success. No error.
EPERM	1	The specified module does not permit the required operation.
EBUSY	12	Device or resource busy. The device is non-responsive or is being accessed by the Link Manager.
EINVAL	14	Invalid argument.

3.7.7.15 Program Topology Device NVM (0x06F2)

This command triggers the write of the external device NVM taken from one of the NVM sections that includes the external topology device image. The programming of the new device firmware into the external device NVM is done using MDIO or I²C commands, depending on the topology device. This admin command is optionally called from the NVM Update Tool. The format of this command is similar to the other topology commands.

3.7.7.15.1 Program Topology Device NVM Command Inputs

Table 3-202 details the input arguments of the Program Topology Device NVM command.

Table 3-202. Program Topology Device NVM Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x06F2	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Logical Port Number	16.[7:0]		Logical Port Number This field specifies the logical port number.
Reserved	17.[7:1]	0x0	Reserved. Must be 0x0.
Logical Port Number Valid	17.[0]		Logical Port Number Valid 0b = Logical port number invalid. 1b = Logical port number valid.
Context	18.[7:4]		Context Software indicates the context within which the handle should be identified. 0 = Global The <i>Index</i> and <i>Node Type</i> are used to identify the node within the context of the entire topology including motherboard and mezzanine cards. Note: This context is not available in this command. EINVAL response is provided if this context is used. 1 = Board The <i>Index</i> and <i>Node Type</i> are used to identify the node within the context of a single board. This could be motherboard or a mezzanine card. Note: This context is not available in this command. EINVAL response is provided if this context is used. 2 = Port The <i>Index</i> and <i>Node Type</i> are used to identify the node within the context of a single port, as specified in the <i>Logical Port Number</i> field. 3 = Node The <i>Index</i> and <i>Node Type</i> are used to identify the node within the context of a single node (i.e. the requested node must be connected in the topology netlist using I/O connection to a specified reference node). Note: This context is not available in this command. EINVAL response is provided if this context is used. 4 = The reference node is provided by software and no search is conducted. Note: This context is not available in this command. EINVAL response is provided if this context is used. All other values are reserved.
Node Type	18.[3:0]		Node Type Software indicates the requested node type.

Table 3-202. Program Topology Device NVM Command Inputs [continued]

Argument	Byte.[Bits]	Value	Description
Index	19.[7:0]		Index For topology searches, software indicates the requested node index. When a PHY device is requested and the context of the search is "Port", PHYs are indexed according to their place in the PHY chain, where 0 means the outermost PHY. For other node types, the nodes are indexed according to their appearance in the netlist. An error is returned when the index is bigger than the number of topology node retrieved in the context search.
Reserved	20:31.[95:0]	0x0	Reserved. Must be 0x0.

3.7.7.15.2 Program Topology Device NVM Command Outputs

Table 3-203 details the output arguments of the Program Topology Device NVM command.

Table 3-203. Program Topology Device NVM Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x06F2	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.7.15.3, "Program Topology Device NVM Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:31.[127:0]	0x0	Reserved. Must be 0x0.

3.7.7.15.3 Program Topology Device NVM Command Errors

The following errors can be returned by the Program Topology Device NVM command:

Table 3-204. Program Topology Device NVM Command Errors

Error Code	Value	Description
OK	0	Success. No error.
ENOENT	2	No such element. For example, node handle was not found. (e.g., Index was higher than the number of handles found for the given context.)
ENXIO	6	No such resource. Firmware did not find a valid external topology device image in the NVM to match the requested device.
EBUSY	12	Device or resource busy. For example, when the command is called when a previous request to program or read the external topology device is ongoing. This code also returned when RDE operation is currently executed.
EINVAL	14	Invalid argument. For example, wrong search context. When using "Node" context, input node handle is not valid.
ENOSYS	17	Device programming not implemented for this device or the device image in NVM is not enabled for programming.

Table 3-204. Program Topology Device NVM Command Errors

Error Code	Value	Description
ERANGE	18	Parameter out of range. For example, when using "Port" context, port number is out of range.
EMODE	21	Operation not allowed in current device mode. For example, when it is called in switch mode or when it is called when link module is disabled or not initialized.

- The device must be in the netlist in order to be accessed. For the typical case of accessing an external PHY, it can be searched by "Port" content. Specifying port number and node type equal to "PHY" and Index of zero provides the outermost PHY. When external PHY is used for multiple ports, any port number that is connected to the external PHY should give the same Node ID.
- Firmware must check that a valid section is present in the NVM with the topology device image. Otherwise, it should return with error (ENXIO - Firmware did not find a valid external topology device image in the NVM to match the requested device).
- Firmware must check that the device part number within the package matches the device pointed to in the topology netlist. Otherwise, it should return with error (ENXIO - Firmware did not find a valid external topology device image in the NVM to match the requested device).
- Firmware must check that the device part number is one of the supported devices that it knows how to program. Otherwise, it should return with error (ENOSYS - Device programming not implemented for this device)

3.7.7.16 Read Topology Device NVM (0x06F3)

This command allows the driver/tool to read the external device NVM. This allows the tool to verify that the external topology device NVM was updated. The format of this command is similar to the other topology commands.

3.7.7.16.1 Read Topology Device NVM Command Inputs

Table 3-202 details the input arguments of the Read Topology Device NVM command.

Table 3-205. Read Topology Device NVM Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.1 for details.
Opcode	2:3.[15:0]	0x06F3	Command opcode.
Datalen	4:5.[15:0]	1-8	Direct command. The size must be 1-8 bytes. Software should not attach buffer. The response is immediate and firmware uses the fields <i>Buffer Address High</i> and <i>Buffer Address Low</i> (bytes 24:31) for the data read from the topology device NVM.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Logical Port Number	16.[7:0]		Logical Port Number This field specifies the logical port number.
Reserved	17.[7:1]	0x0	Reserved. Must be 0x0.
Logical Port Number Valid	17.[0]		Logical Port Number Valid 0b = Logical port number invalid. 1b = Logical port number valid.
Context	18.[7:4]		Context Software indicates the context within which the handle should be identified. 0 = Global The <i>Index</i> and <i>Node Type</i> are used to identify the node within the context of the entire topology including motherboard and mezzanine cards. Note: This context is not available in this command. EINVAL response is provided if this context is used. 1 = Board The <i>Index</i> and <i>Node Type</i> are used to identify the node within the context of a single board. This could be motherboard or a mezzanine card. Note: This context is not available in this command. EINVAL response is provided if this context is used. 2 = Port The <i>Index</i> and <i>Node Type</i> are used to identify the node within the context of a single port, as specified in the <i>Logical Port Number</i> field. 3 = Node The <i>Index</i> and <i>Node Type</i> are used to identify the node within the context of a single node (i.e. the requested node must be connected in the topology netlist using I/O connection to a specified reference node). Note: This context is not available in this command. EINVAL response is provided if this context is used. 4 = The reference node is provided by software and no search is conducted. Note: This context is not available in this command. EINVAL response is provided if this context is used. All other values are reserved.
Node Type	18.[3:0]		Node Type Software indicates the requested node type.

Table 3-205. Read Topology Device NVM Command Inputs [continued]

Argument	Byte.[Bits]	Value	Description
Index	19.[7:0]		Index For topology searches, software indicates the requested node index. When a PHY device is requested and the context of the search is "Port", PHYs are indexed according to their place in the PHY chain, where 0 means the outermost PHY. For other node types, the nodes are indexed according to their appearance in the netlist. An error is returned when the index is bigger than the number of topology node retrieved in the context search.
Start Address	20:23.[31:0]		Start Address The start address (in bytes) in the topology device NVM.
Data Read	24:31.[63:0]		Data Read Data bytes as read from the topology device NVM.

3.7.7.16.2 Read Topology Device NVM Command Outputs

Table 3-206 details the output arguments of the Read Topology Device NVM command.

Table 3-206. Read Topology Device NVM Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.1.2 for details.
Opcode	2:3.[15:0]	0x06F3	Command opcode.
Datalen	4:5.[15:0]	0x0	Length of the data buffer in bytes.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.7.16.3, "Read Topology Device NVM Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:19.[31:0]	0x0	Reserved. Must be 0x0.
Start Address	20:23.[31:0]		Start Address The start address (in bytes) in the topology device NVM.
Data Read	24:31.[63:0]		Data Read Data bytes as read from the topology device NVM.

3.7.7.16.3 Read Topology Device NVM Command Errors

The following errors can be returned by the Read Topology Device NVM command:

Table 3-207. Read Topology Device NVM Command Errors

Error Code	Value	Description
OK	0	Success. No error.
ENOENT	2	No such element. For example, node handle was not found. (e.g., Index was higher than the number of handles found for the given context.)
EIO	5	Read Return with an error.
EBUSY	12	Device or resource busy. For example, when the command is called when a previous request to program or read the external topology device is ongoing. This code also returned when RDE operation is currently executed.

Table 3-207. Read Topology Device NVM Command Errors [continued]

Error Code	Value	Description
EINVAL	14	Invalid argument. For example, wrong search context. When using "Node" context, input node handle is not valid.
ENOSYS	17	Device programming not implemented for this device or the device image in NVM is not enabled for programming.
EMODE	21	Operation not allowed in current device mode. For example, when it is called in switch mode or when it is called when link module is disabled or not initialized.

Notes:

- The device must be in the netlist in order to be accessed. For the typical case of accessing an external PHY, it can be searched by "Port" content. Specifying port number and node type equal to "PHY" and Index of zero provides the outermost PHY. When external PHY is used for multiple ports, any port number that is connected to the external PHY should give the same Node ID.
- This command is direct (i.e. without buffer). Software can read 1-8 bytes without buffer and the response is immediate. When software needs to read more bytes, it must call this command again with a different address to read additional bytes. This command is blocked during external topology device programming. The command is available when the external topology device programming is done.

3.7.8 DNL Engine

The Link Manager instantiates a script interpreter engine dedicated for link management purposes. The engine conforms to the Dreadnought Lake 3.3 specification.

3.7.8.1 DNL Topology Access

The Dreadnought Lake script engine provides CSR register access to the device's BAR0 internal registers through the READCSR and WRITECSR actions. Additional CSRs of devices that are connected to the E610 via IOSF-SB can be accessed through the READIOSF and WRITEIOSF actions. All CSR registers are readable but only a controlled section can be modified by DNL scripts. For more detail, refer to [Section 3.8.7, "Link Topology Access Allowlist"](#).

3.7.8.2 DNL Script Library

The Dreadnought Lake scripts, also referred to as activities, called by the Link Management application are defined by the Dreadnought Lake 3.4 specification.

These scripts, in conjunction with a set of user defined scripts that form the Dreadnought Lake Script library, are stored in binary format in the signed part of the NVM. The E610 NVM allocates 320 KB for this section.

3.7.9 DNL Engine Commands

3.7.9.1 DNL Commands Summary

Table 3-208 summarizes the commands that are part of the Dreadnought Lake 3.4 development and debug environment.

Table 3-208. DNL Commands

Command	Opcode	Description	Section Reference
DNL_CALL	0x682	Used to call a DNL activity.	Section 3.7.9.2

Note: All Dreadnought Lake Commands require Debug mode to be enabled. The only exception is DNL_CALL with SET_DFX_COMMAND, which is allowed in any mode.

Note: All reserved fields within the command arguments must be 0.

3.7.9.2 DNL_CALL (0x682)

The DNL_CALL command used to call a DNL activity in the specified context. The input arguments are taken from STO_0, STO_1, STO_3, and STO_4. The command completes once the called activity returns. The called activity's return values are returned through the same STO_0, STO_1, STO_3, and STO_4.

This command is allowed only when the DNL engine is idle. Issuing DNL_CALL while the engine is not idle results in the DNL engine returning EMODE.

This command can call any DNL activity when host debug mode is enabled. When host debug mode is not enabled, the DNL engine returns EMODE for all activities. The only exception is DNL_CALL with SET_DFX_COMMAND, which is allowed in any mode.

3.7.9.2.1 DNL_CALL Command Inputs

Table 3-209 details the input arguments of the DNL_CALL command.

Table 3-209. DNL_CALL Command Inputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.2.1 for details.
Opcode	2:3.[15:0]	0x682	Command opcode.
Datalen	4:5.[15:0]	16	Length in bytes of the command buffer.
Return Value/VFID	6:7.[15:0]		Return value. Zeroed by driver.
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16.[7:5]	000b	Reserved. Must be 000b.
Context	16.[4:0]		Context The context in which the activity should be executed. 0x00:0x07 = For activities needing to run in the context of a port, the context matches the port number. 0x1E = For activities needing to run in the global context, the context is 0x1E. All other values are reserved.
Reserved	17.[7:0]	0x0	Reserved. Must be 0x0.
Activity ID	18:19.[15:0]		Activity ID The ID of the activity to be called.
Reserved	20:23.[31:0]	0x0	Reserved. Must be 0x0.
Data Address High	24:27.[31:0]		High bits of command buffer address.
Data Address Low	28:31.[31:0]		Low bits of command buffer address.

Table 3-210. DNL_CALL Command Input Buffer

Argument	Byte.[Bits]	Value	Description
STO_0	4:7.[31:0]		
STO_1	8:11.[31:0]		
STO_2	12:15.[31:0]		
STO_3	16:19.[31:0]		

3.7.9.2.2 DNL_CALL Command Outputs

Table 3-211 details the output arguments of the DNL_CALL command.

Table 3-211. DNL_CALL Command Outputs

Argument	Byte.[Bits]	Value	Description
Flags	0:1.[15:0]		See Section 10.11.2.4.2.2 for details.
Opcode	2:3.[15:0]	0x682	Command opcode.
Datalen	4:5.[15:0]	16	Length in bytes of the command buffer.
Return Value/VFID	6:7.[15:0]		Return value. Written by firmware. See Section 3.7.9.2.3, "DNL_CALL Command Errors" .
Cookie High	8:11.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12:15.[31:0]	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16:17.[15:0]	0x0	Reserved. Must be 0x0.
Activity ID	18:19.[15:0]		Activity ID The ID of the activity to be called.
Reserved	20:23.[31:0]	0x0	Reserved. Must be 0x0.
Data Address High	24:27.[31:0]		High bits of command buffer address.
Data Address Low	28:31.[31:0]		Low bits of command buffer address.

Table 3-212. DNL_CALL Command Output Buffer

Argument	Byte.[Bits]	Description
STO_0	0:3.[31:0]	
STO_1	4:7.[31:0]	
STO_2	8:11.[31:0]	
STO_3	12:15.[31:0]	

3.7.9.2.3 DNL_CALL Command Errors

The following errors can be returned by the DNL_CALL command:

Table 3-213. DNL_CALL Command Errors

Error Code	Value	Description
OK	0	No error
ENOENT	2	No such element. No activity matching opcode was found.
EFAULT	11	Internal firmware error.
EINVAL	14	Invalid argument. Invalid context (0x1F).
EMODE	21	Operation not allowed in current device mode. Host debug mode disabled.

3.8 Link Topology

3.8.1 Overview

The E610 makes use of a configurable link management engine referred to as the Link Manager. Configuring Ethernet links over various link topologies requires this engine to be aware of the components that are part of the link topology, the connections between these components, including both high and low speed connections.

The scheme of the components involved associated configuration and connections is referred to as Ethernet link topology. The link topology includes the integrated PHYs, integrated GPIO controller (i.e., within the device), and might include external PHYs, re-timers, cages for pluggable media (i.e SFP cage), LEDs and temperature sensors.

The high-speed connections between the PHYs and cages allows the Link Manager to identify all link segments to configure for a particular port. The associated meta data is used to define the properties of these link segments also referred to as PHY Capabilities.

Some of the components included in the Ethernet link topology require configuration via register writes while others are configured through sideband signals. The link topology provides the Link Manager with the necessary information on which bus and at what bus address to access these components. The integrated PHYs can be accessed using buses internal to the device, however externally connected devices are usually accessible through I²C or MDIO. For I²C buses, the link topology allows for use of I²C muxes to expand the I²C tree. These are configured by the Link Manager as needed to access specific devices on the I²C tree.

The link topology also provides description of specific sideband signals such as RESET_N, PRESENT_N and INT_N. Either directly connected to the integrated GPIO controller or indirectly to a GPIO controller accessible on the I²C tree, these signals are configured and monitored by the Link Manager to detect changes in the link topology such as modules inserted or removed.

Finally, the link management task also includes control of link related LEDs. The link topology provides the Link Manager with the connections and configuration of these LEDs. These may be monochrome LEDs driven directly by pins of the integrated GPIO controller or an I²C attached GPIO or dedicated LED controller, or they may be RGB LEDs driven by an I²C attached dedicated LED controller.

The link topology of the board holding the device is stored in the Flash the device is booted from. During initialization, the Link Manager reads this link topology and initialize all devices accordingly.

Some parts of the link topology might change dynamically. For example, a pluggable module (i.e. SFP module) is inserted or removed from a specific cage. This hot-plug event is captured by the Link Manager and the link topology is configured appropriately.

This section presents the structures describing the above introduced link topology and details the flows followed by the Link Manager to read, process and configure the topology as well as describes the software interface available to access the link topology.

3.8.2 Link Topology Definition

The link topology describes any devices connected to the Ethernet link side. It includes the integrated PHYs, integrated GPIO controller (i.e. within the device), and might include external PHYs, re-timers, cages for pluggable media (i.e. SFP), LEDs, and temperature sensors.

There are 10 cases of E610 topology netlist, as detailed in [Table 3-214](#).

Table 3-214. E610 Topology Netlist Cases

Package Type	Case	Port Mode		Included in the Topology Netlist	
		Port 0	Port 1	Port 0	Port 1
E610 MCP	0	1GBASE-T	1GBASE-T	Everything just like CVL, MEV, MEV-TS, and CNV.	
E610 SCP	1	C2M	C2M	1. E610 2. 4 LED pins per port 3. 4 dedicated pins over GPIO (Present_N, Tx_Disable, RS0/RS1, RX_LOS) per port 4. I²C over GPIO for DPLL 5. I2C0/I2C1 for SFP 6. 1 SFP cage per port	
	2	C2C	C2C	1. E610 2. 4 LED pins per port 3. 4 dedicated pins over GPIO (customer Define in NVM) per port 4. I²C over GPIO for DPLL 5. MDIO0/MDIO1 for external PHY	
	3	KR	KR	1. E610 2. 4 LED pins per port 3. 4 dedicated pins over GPIO (customer Define in NVM) per port 4. I²C over GPIO for DPLL	
	4	C2M	C2C	Same as case 1 for Port 0.	Same as case 2 for Port 1.
	5	C2C	C2M	Same as case 2 for Port 0.	Same as case 1 for Port 1.
	6	C2M	KR	Same as case 1 for Port 0.	Same as case 3 for Port 1.
	7	KR	C2M	Same as case 3 for Port 0.	Same as case 1 for Port 1.
	8	C2C	KR	Same as case 2 for Port 0.	Same as case 3 for Port 1.
	9	KR	C2C	Same as case 3 for Port 0.	Same as case 2 for Port 1.

[Figure 3-39](#) show a generic example of what's included in the link topology. [Figure 3-40](#) through [Figure 3-49](#) show the diagrams for each of the topology netlist cases detailed in [Table 3-214](#).

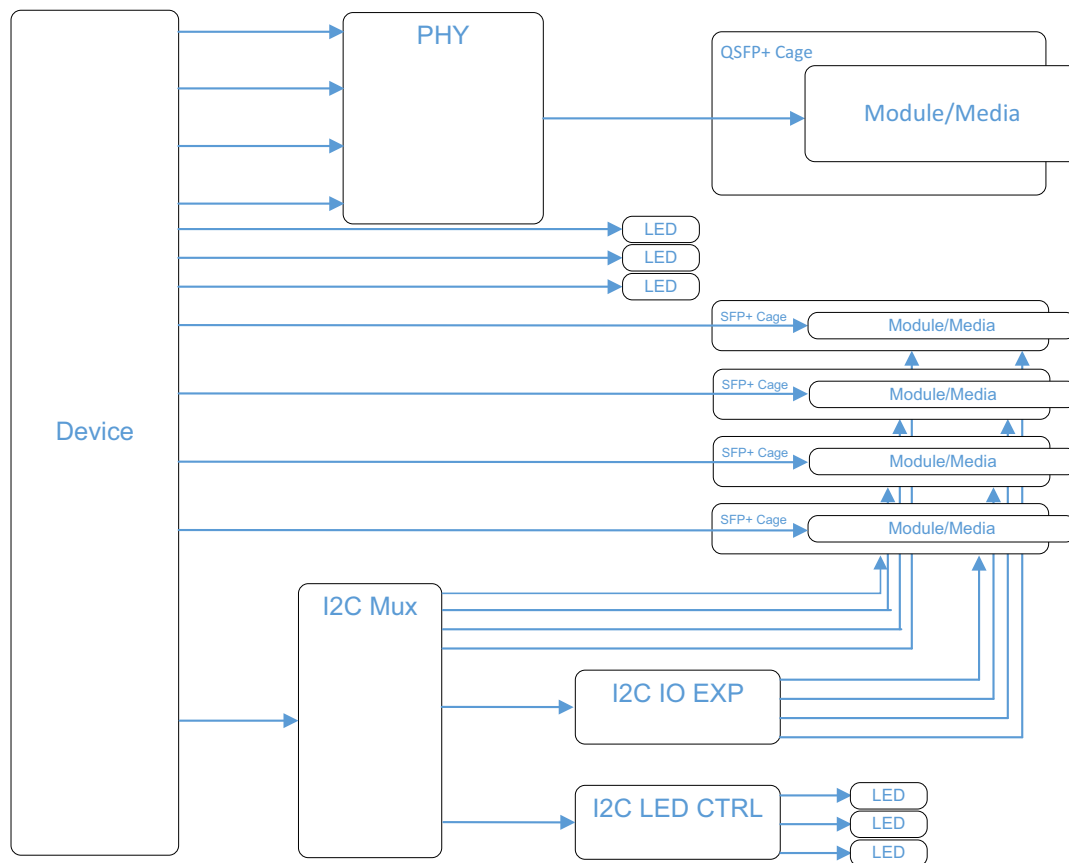


Figure 3-39. Generic Link Topology Example

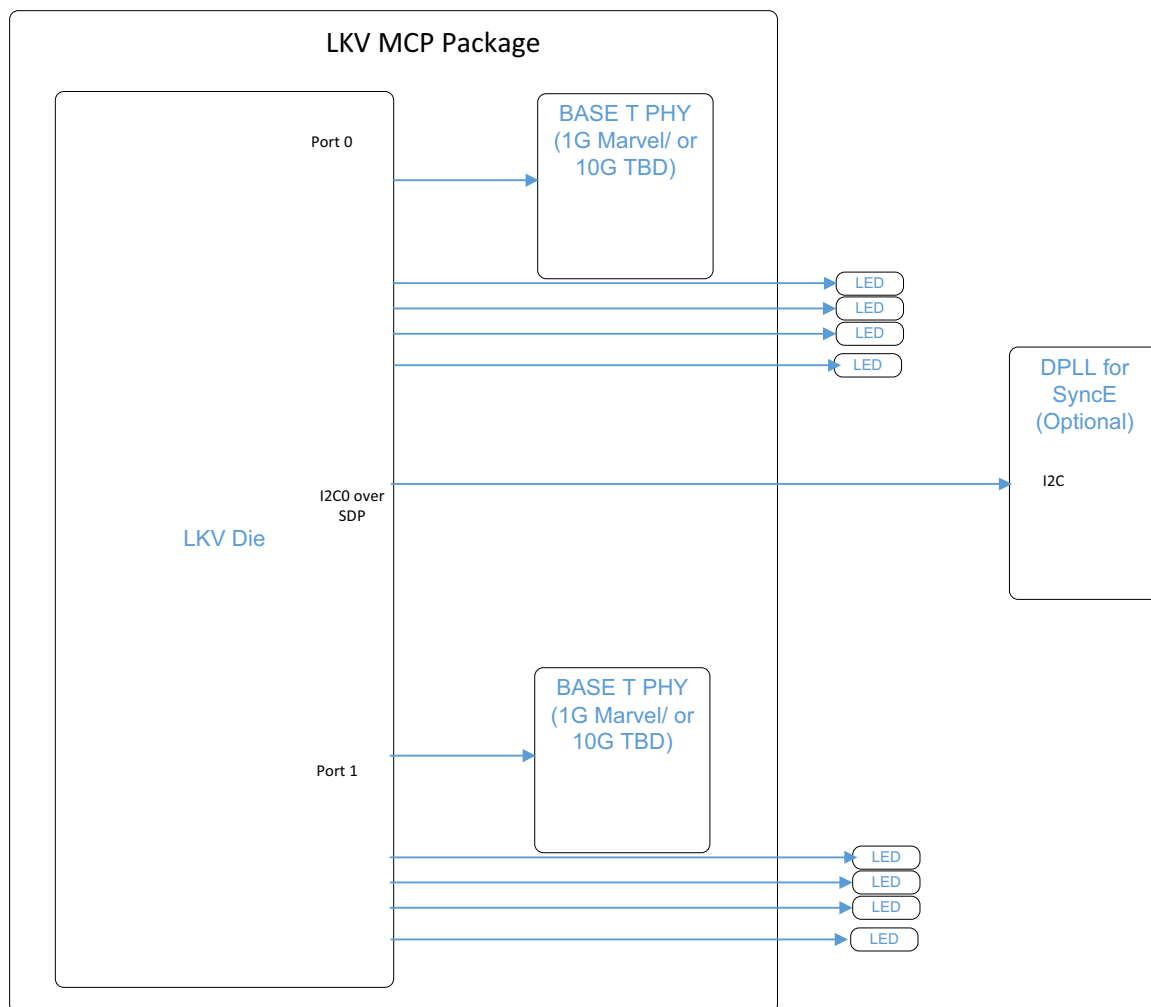


Figure 3-40. Topology Netlist - Case 0 (E610 MCP)

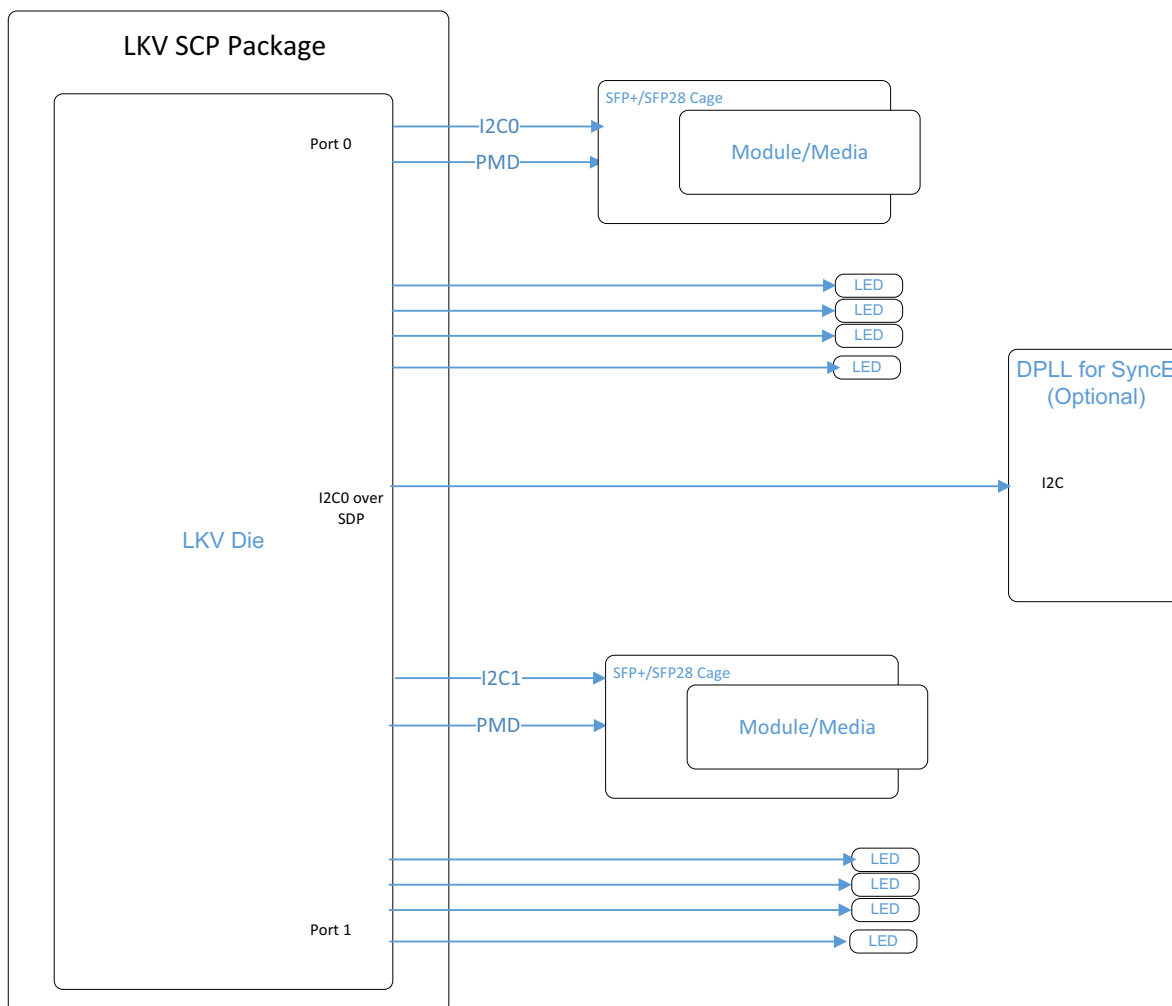


Figure 3-41. Topology Netlist - Case 1 (E610 SCP - C2M/C2M)

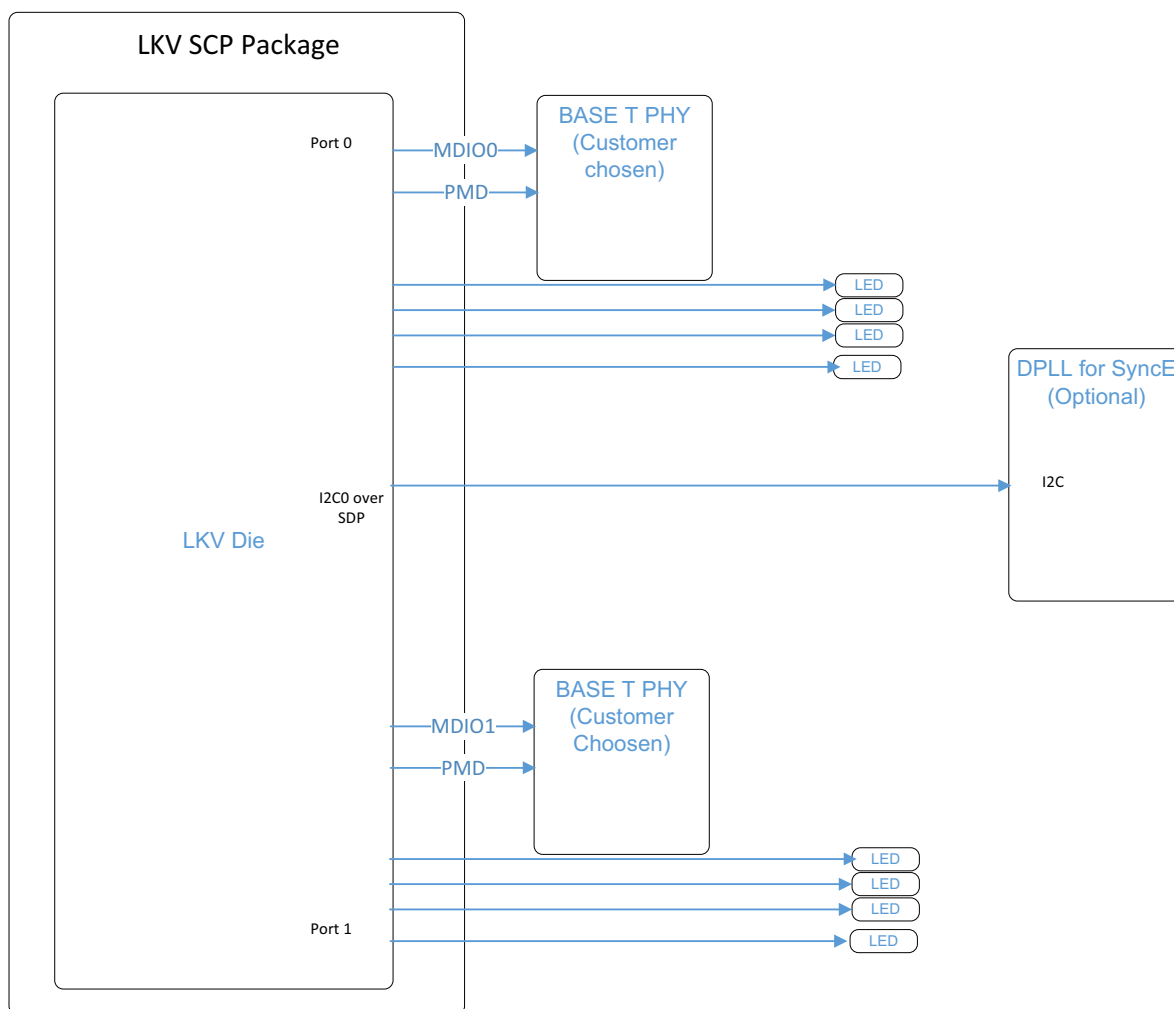


Figure 3-42. Topology Netlist - Case 2 (E610 SCP - C2C/C2C)

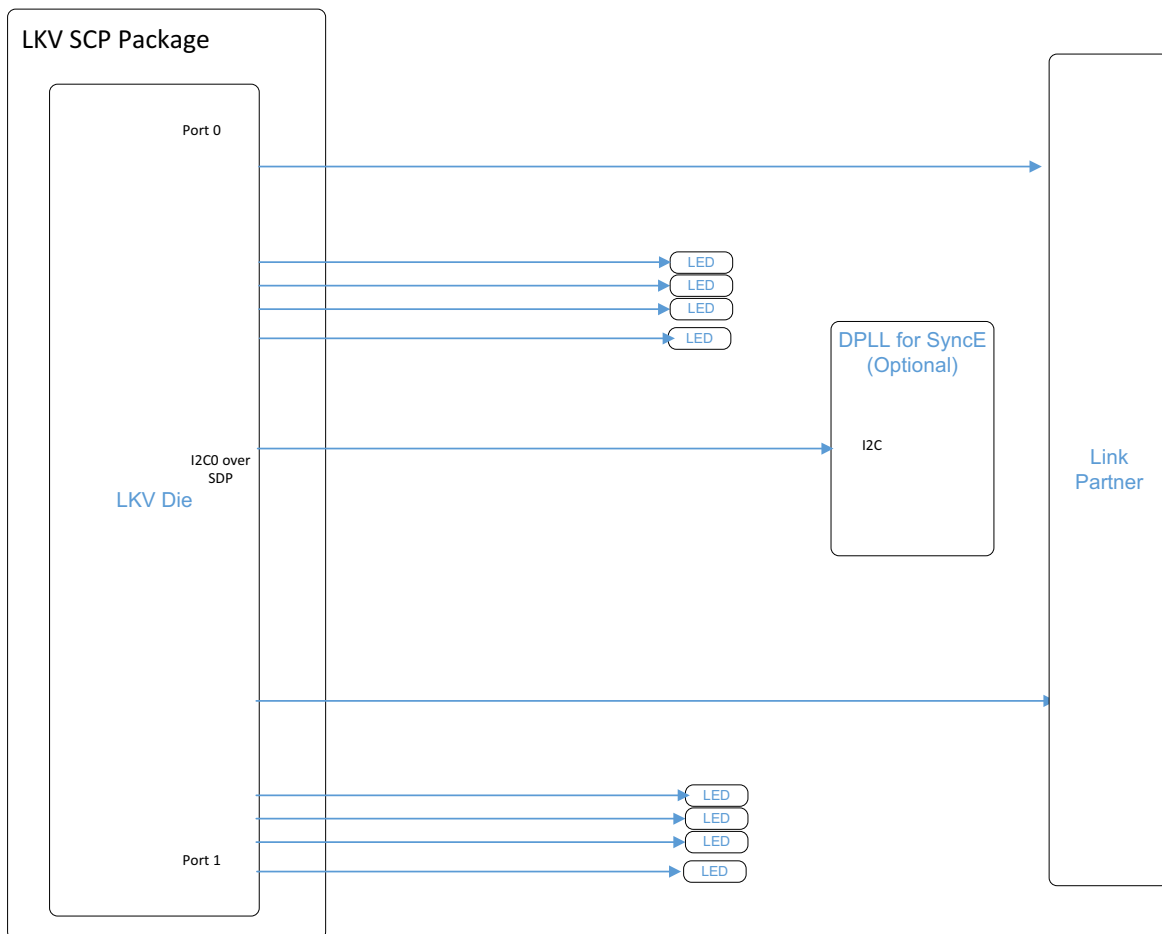


Figure 3-43. Topology Netlist - Case 3 (E610 SCP - KR/KR)

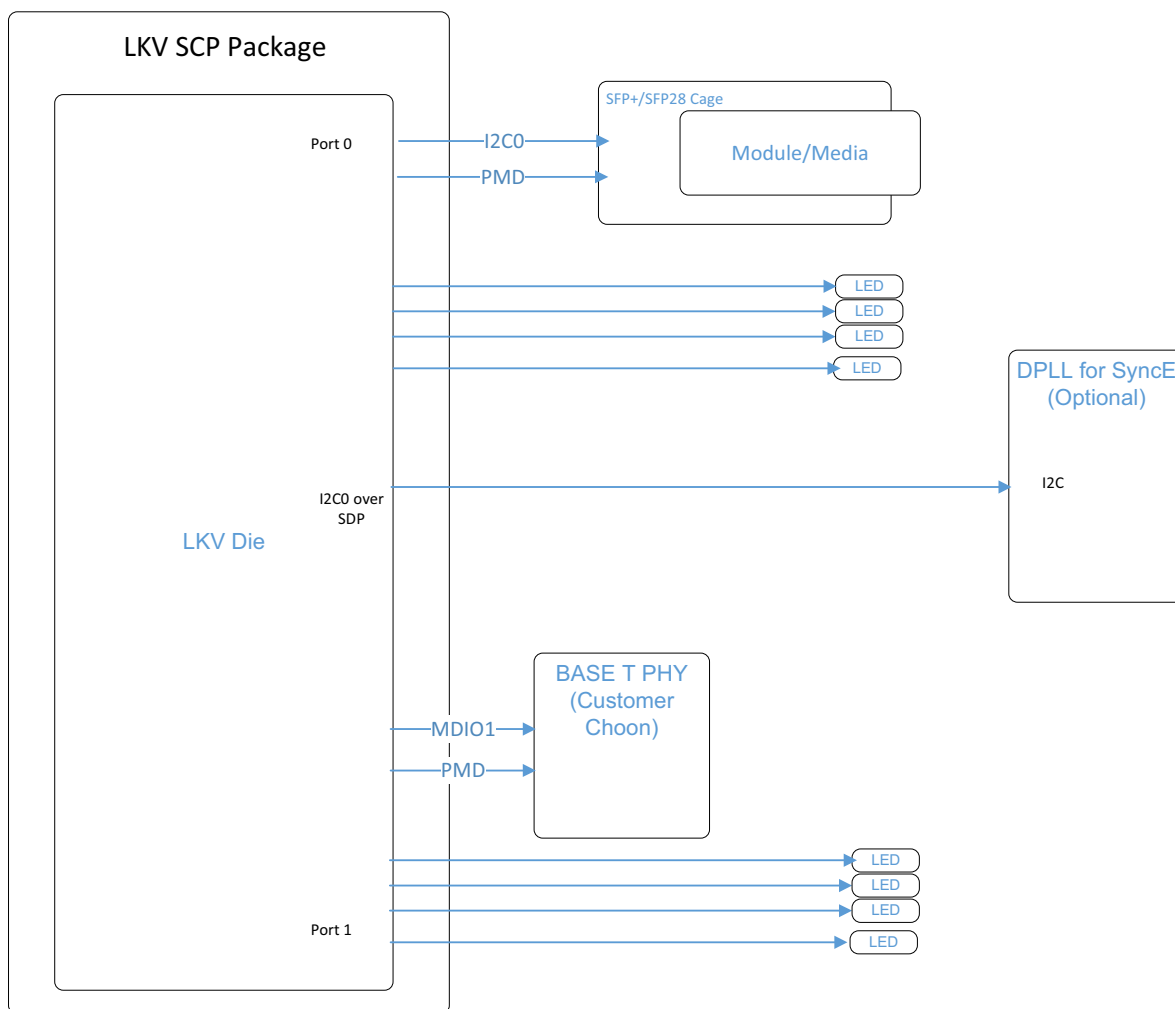


Figure 3-44. Topology Netlist - Case 4 (E610 SCP - C2M/C2C)

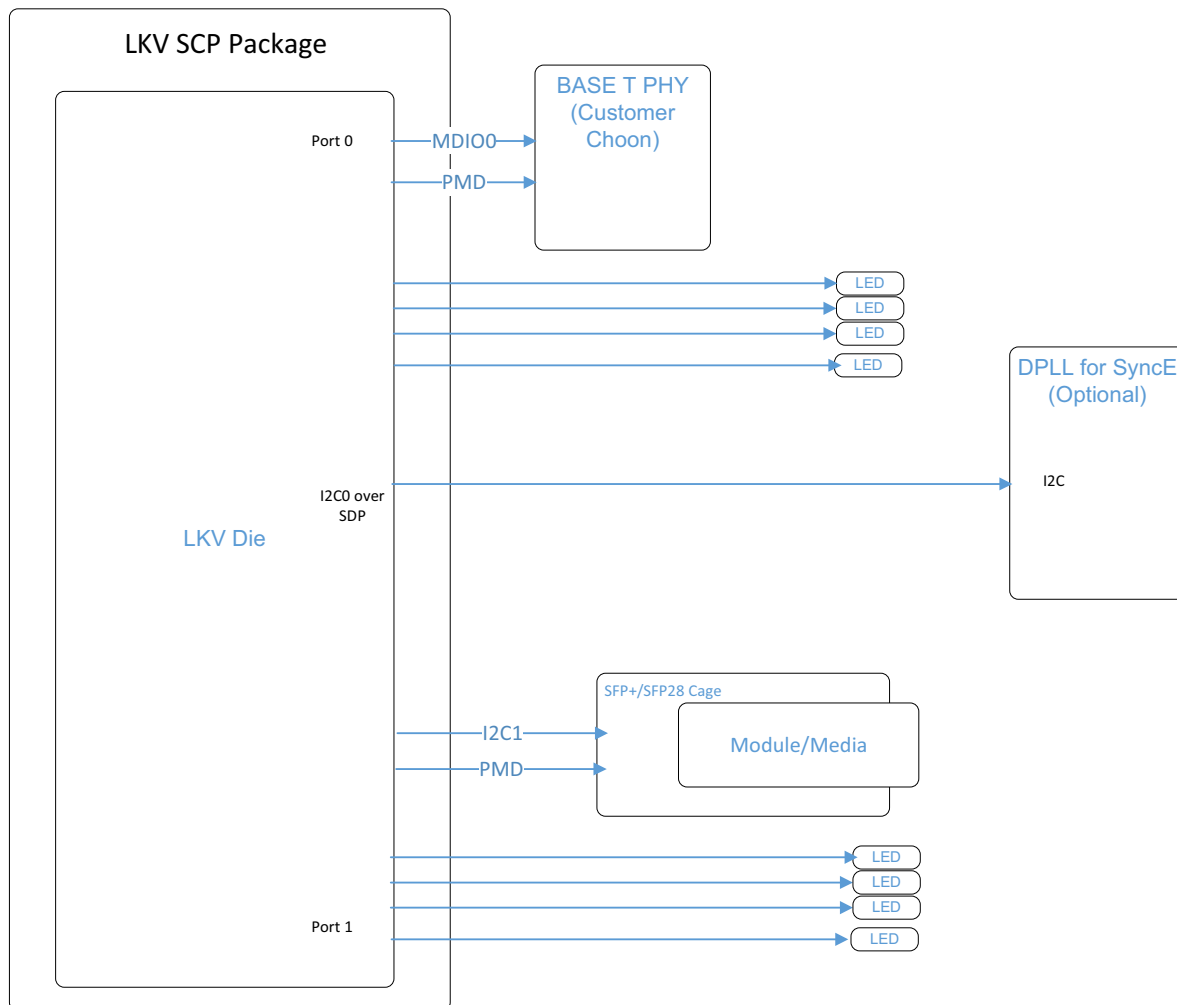


Figure 3-45. Topology Netlist - Case 5 (E610 SCP - C2C/C2M)

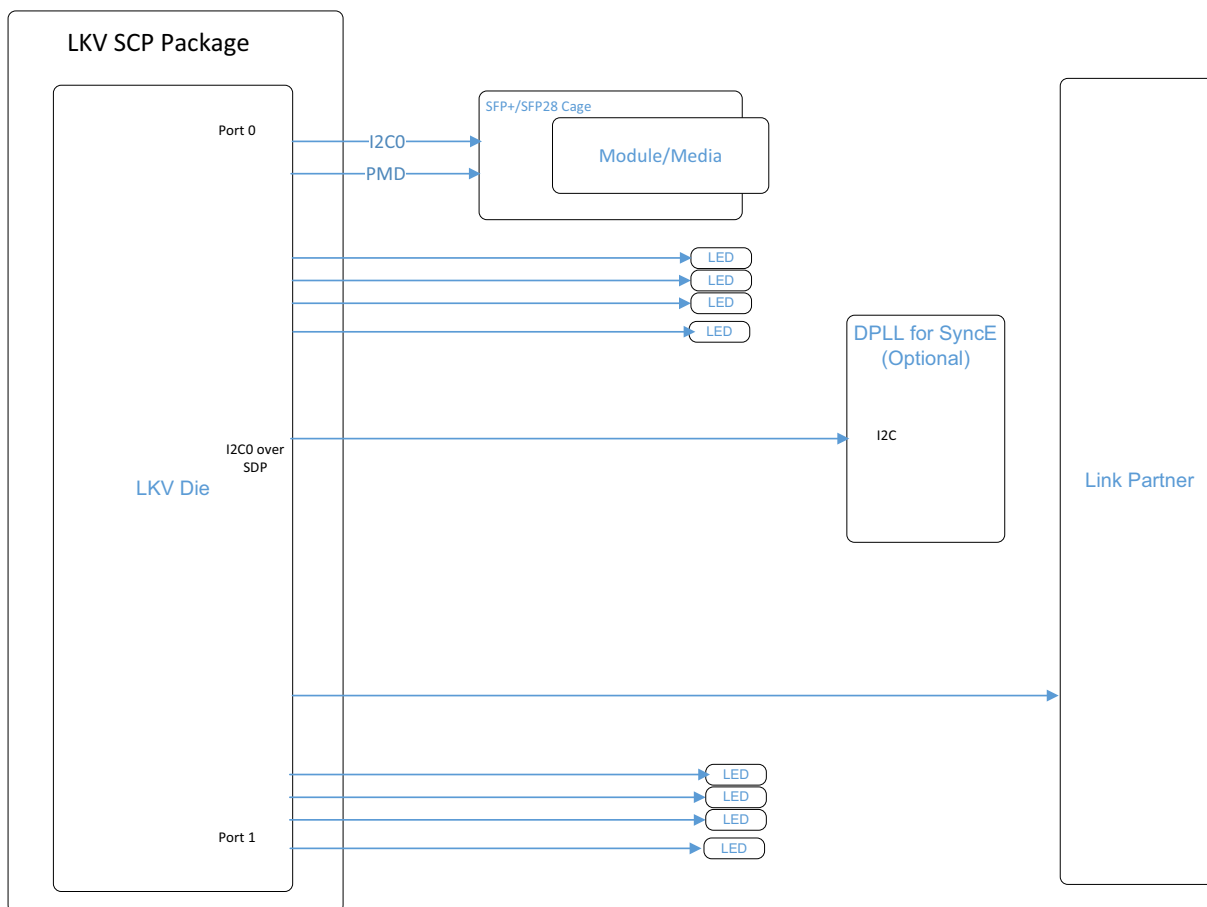


Figure 3-46. Topology Netlist - Case 6 (E610 SCP - C2M/KR)

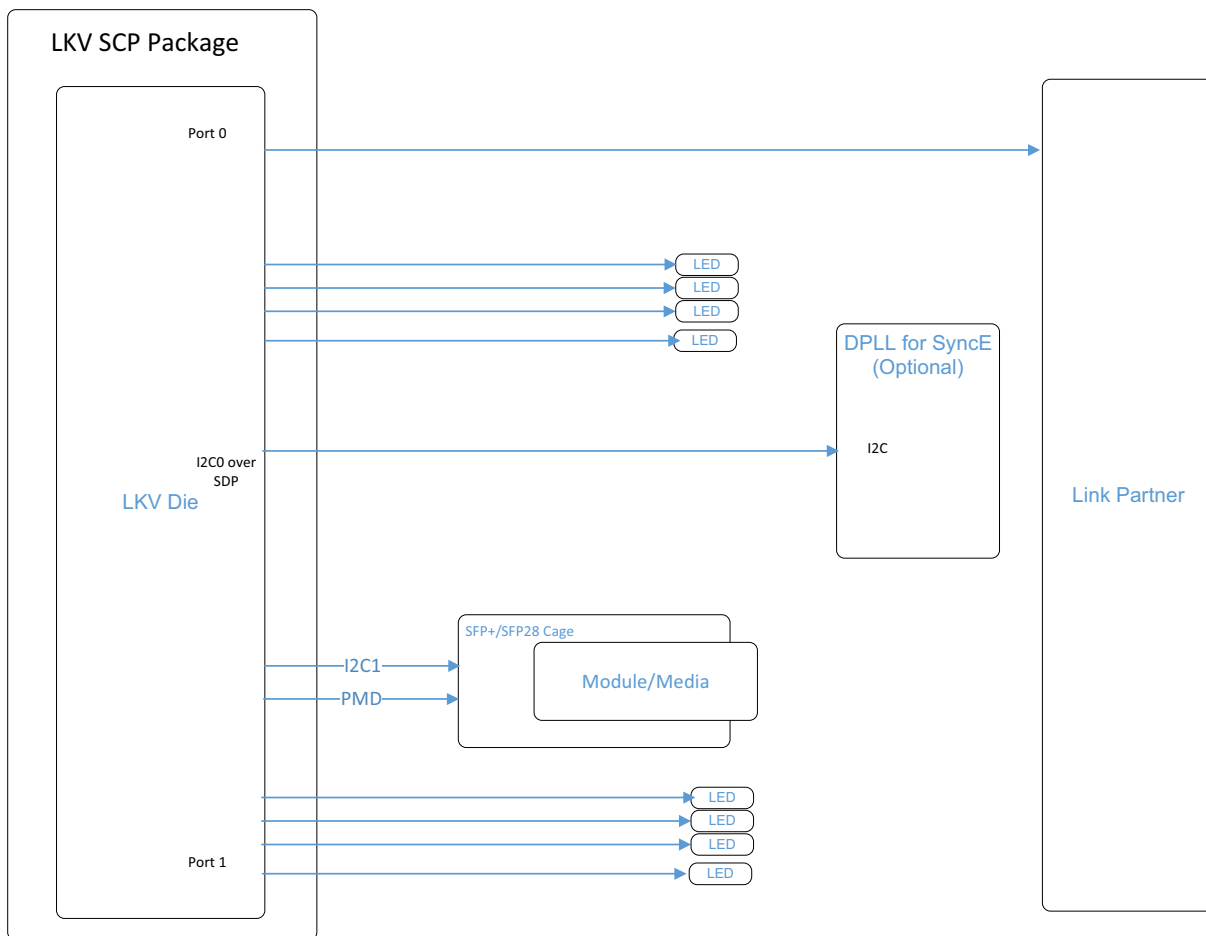


Figure 3-47. Topology Netlist - Case 7 (E610 SCP - KR/C2M)

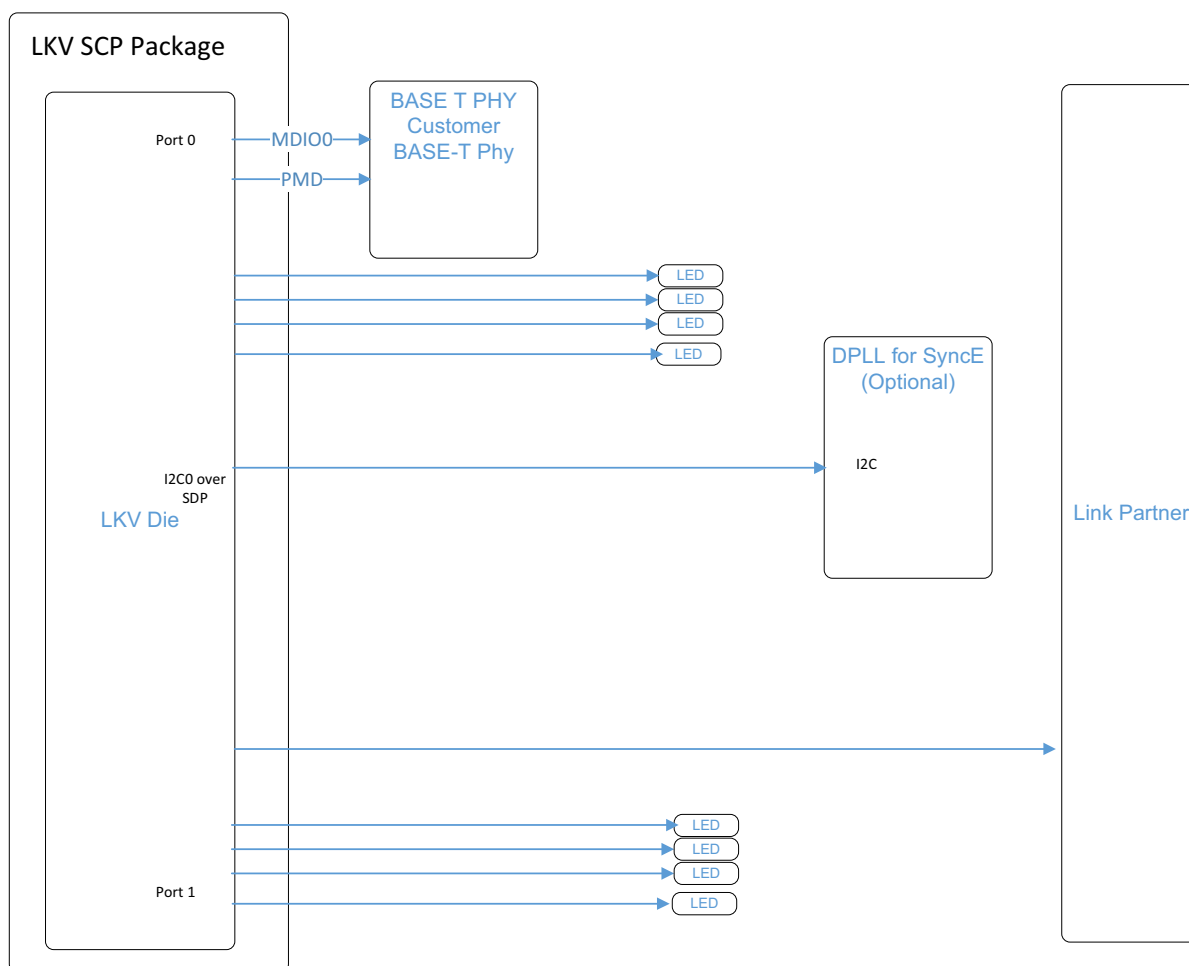


Figure 3-48. Topology Netlist - Case 8 (E610 SCP - C2C/KR)

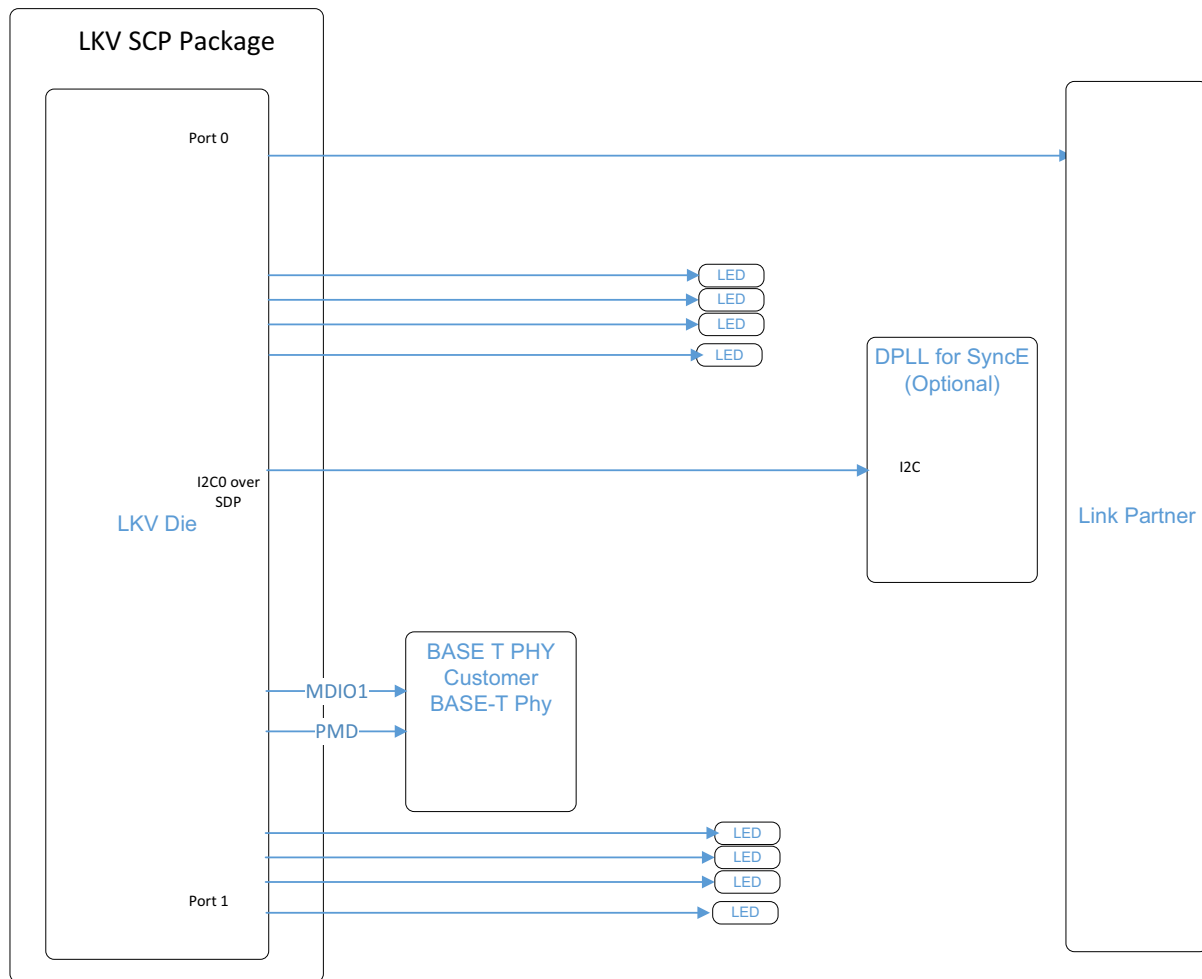


Figure 3-49. Topology Netlist - Case 9 (E610 SCP - KR/C2C)

3.8.2.1 SDP Allocation for Various E610 SKUs

For E610 MCP with 1GBASE-T or 10GBASE-T SKU, the SDP allocation is as detailed in [Table 3-215](#).

Table 3-215. On E610 Die in E610 MCP with 1GBASE-T 10GBASE-T

	PAD	Group	Type	1GBASE-T Port 0	1GBASE-T Port 1	Global	Notes
Without SyncE	SDP0	LED0	Output	LKP_Leds_0[0]			
	SDP1	LED0	Output	LKP_Leds_0[1]			
	SDP2	LED0	Output	LKP_Leds_0[2]			
	SDP3	LED0	Output	LKP_Leds_0[3]			
	SDP4	LED1	Output		LKP_Leds_1[0]		
	SDP5	LED1	Output		LKP_Leds_1[1]		
	SDP6	LED1	Output		LKP_Leds_1[2]		
	SDP7	LED1	Output		LKP_Leds_1[3]		
	I2C_MDIO_CLK0	BASE-T PHY0	Output	I2C_MDIO_CLK0			
	I2C_MDIO_DATA0	BASE-T PHY0	I/O	I2C_MDIO_DATA0			
	SDP24	BASE-T PHY0	Output	RESET0n			
	SDP26	BASE-T PHY0	Input	INT0_N			
	I2C_MDIO_CLK1	BASE-T PHY1	Output		I2C_MDIO_CLK1		
	I2C_MDIO_DATA1	BASE-T PHY1	I/O		I2C_MDIO_DATA1		
	SDP25	BASE-T PHY1	Output		RESET1n		
	SDP27	BASE-T PHY1	Input		INT1_N		
	ETH_CLK_P	CLOCK	Input			ETH_CLK_P	Assumes 156.25 diff from DPLL. This is POR for now.
	ETH_CLK_N	CLOCK	Input			ETH_CLK_N	
With SyncE	SDP18	DPLL	Output			I2C_DPLL_SCL	
	SDP19	DPLL	I/O			I2C_DPLL_SDA	
	N/A	DPLL	Input			INT_N	Not connected in Westport Channel (OUT6P)
	N/A	DPLL	Output			Reset_N	From Board POR logic.
	SDP22	GPS	Output			I2C_GPS_SCL	Westport Channel is using separate I ² C pins than the DPLL I ² C pins.
	SDP23	GPS	I/O			I2C_GPS_SDA	
	N/A	GPS	Input			Reset_N	Goes to I/O Expander.
	N/A	GPS	?			Safe_Boot_N	Goes to I/O Expander.
	N/A	GPS	?			Int_N	Goes to I/O Expander.
	N/A	GPS	Input			Present_N	Goes to I/O Expander.
	N/A	GPS	Output			CLK_OUT_0	Goes to DPLL CLK_IN.
	N/A	GPS	Output			CLK_OUT_1	Not used in Westport Channel. So, keep it unused.
	N/A	I/O Expander	Output			I2C_GPS_SCL	Same I ² C as GPS.
	N/A	I/O Expander	I/O			I2C_GPS_SDA	Same I ² C as GPS.
	SDP10	I/O Expander	Output			Int_N	
	SDP11	I/O Expander	Output			RESER_N	CVL, ETT netlist shows unconnected.

For E610 SCP with SFP on both ports, the SDP allocation is as detailed in [Table 3-216](#).

Table 3-216. On E610 Die in E610 SCP with SFP

	PAD	Group	Type	1GBASE-T Port 0	1GBASE-T Port 1	Global	Notes
Without SyncE	SDP0	LED0	Output	LKP_Leds_0[0]			
	SDP1	LED0	Output	LKP_Leds_0[1]			
	SDP2	LED0	Output	LKP_Leds_0[2]			
	SDP3	LED0	Output	LKP_Leds_0[3]			
	SDP4	LED1	Output		LKP_Leds_1[0]		
	SDP5	LED1	Output		LKP_Leds_1[1]		
	SDP6	LED1	Output		LKP_Leds_1[2]		
	SDP7	LED1	Output		LKP_Leds_1[3]		
	I2C_MDIO_CLK0	SFP_0	Output	I2C_MDIO_CLK0			
	I2C_MDIO_DATA0	SFP_0	I/O	I2C_MDIO_DATA0			
	SDP8	SFP_0	Output	SFP0 TX_DISABLE			When SyncE is enabled, these 4 pins can go to I/O Expander.
	SDP9	SFP_0	Input	SFP0 MOD_ABS			
	SDP10	SFP_0	Input	SFP0 RX_LOS			
	SDP11	SFP_0	Output	SFP0 RS0/RS1			
	I2C_MDIO_CLK1	SFP_1	Output		I2C_MDIO_CLK1		
	I2C_MDIO_DATA1	SFP_1	I/O		I2C_MDIO_DATA1		
	SDP12	SFP_1	Output		SFP1 TX_DISABLE		When SyncE is enabled, these 4 pins can go to I/O Expander.
	SDP13	SFP_1	Input		SFP1 MOD_ABS		
	SDP14	SFP_1	Input		SFP1 RX_LOS		
	SDP15	SFP_1	Output		SFP1 RS0/RS1		
	ETH_CLK_P	CLOCK	Input			ETH_CLK_P	Assumes 156.25 diff from DPLL.
	ETH_CLK_N	CLOCK	Input			ETH_CLK_N	

Table 3-216. On E610 Die in E610 SCP with SFP [continued]

	PAD	Group	Type	1GBASE-T Port 0	1GBASE-T Port 1	Global	Notes
With SyncE		DPLL	Output			I2C_DPLL_SCL	
		DPLL	I/O			I2C_DPLL_SDA	
	N/A	DPLL	Input			INT_N	Not connected in Westport Channel (OUT6P)
	N/A	DPLL	Output			Reset_N	From Board POR logic.
		GPS	Output			I2C_GPS_SCL	Westport Channel is using separate I ² C pins than the DPLL I ² C pins.
		GPS	I/O			I2C_GPS_SDA	
	N/A	GPS	Input			Reset_N	Goes to I/O Expander.
	N/A	GPS	?			Safe_Boot_N	Goes to I/O Expander.
	N/A	GPS	?			Int_N	Goes to I/O Expander.
	N/A	GPS	Input			Present_N	Goes to I/O Expander.
	N/A	GPS	Output			CLK_OUT_0	Goes to DPLL CLK_IN.
	N/A	GPS	Output			CLK_OUT_1	Not used in Westport Channel. So, keep it unused.
	N/A	I/O Expander	Output			I2C_GPS_SCL	Same I ² C as GPS.
	N/A	I/O Expander	I/O			I2C_GPS_SDA	Same I ² C as GPS.
	SDP26	I/O Expander	Output			Int_N	
	SDP24	I/O Expander	Output			RESER_N	Diagram shows connected to CVL, ETT netlist shows unconnected.
	SDP8	CLOCK	Output			SYNCE_CLKOUT_0	
	SDP9	CLOCK	Output			SYNCE_CLKOUT_1	

3.8.2.2 High-Speed PHY Chain

One part of the link topology can be viewed as a chain of PHYs, re-timers, and pluggable modules that are connected together to perform a link. The PHY nodes are considered to have host-side and line-side interfaces with one exception where the integrated PHY only has a line-side interface. The line-side interface of a node connected to the host-side interface of other forms an internal link segment. The line-side interface of the outermost node interfacing with the link partner forms the outer link segment.

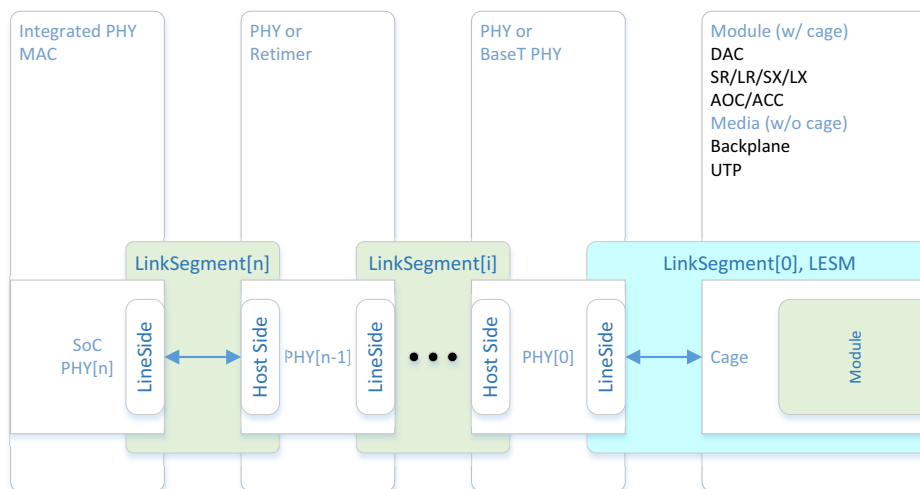


Figure 3-50. High-Speed PHY Chain

Notes:

- Some components in the PHY chain are within the device, while others are on the board/
- Some topologies do not include all components (e.g. re-timer or a cage).
- The high-speed PHY chain might contain components that are shared between several Ethernet links. For example, an external PHY might support four Ethernet links.

The capabilities of each of these link segments is defined through the PHY capabilities structures and serves as configuration information for the Link Establishment State Machine (LESM)

3.8.3 Topology Structures

The link topology of a given board is described using a link topology netlist. This describes all topology components, the connection between them, the parameters of these connections and the relevant component specific configuration information.

The netlist corresponding to the Network Interface Card (NIC)/ Local Area Network (LAN) on Motherboard (LOM) link topology is stored in the Flash the device is booted from. This includes all components and connections within the NIC/LOM.

3.8.3.1 PHY Capabilities

The Ethernet link can support different speeds and different electrical parameters. While a given link mode might have additional options described in adjacent parameters, the combination of a certain speed and a specific PMD is referred to as PHY Type. The list of the supported PHY Types forms what is referred to the PHY Capabilities structure.

Table 3-217 shows the list of the PHY Types supported by the E610 along with the corresponding media, auto-negotiation and FEC configuration options, and uses the following definitions:

- R - Required
- S - Supported
- O - Optional
- U - Unsupported (but may be enabled through AutoFEC)
- V - Variable

Table 3-217. Supported PHY Types

Index	Name	Auto-Neg			Media				FEC Type		Cage Type	PMD	POR
		C128	C137	C173	Optics/Active	DAC	Back-plane	Twisted-pair	10G KR	No FEC	SFP	Width	
0x0	100BASE-Tx							R				x1	S
0x1	100SGMII		R									x1	S
0x2	1000BASE-T	R						R		S	S	x1	S
0x3	1000BASE-SX				R					S	S	x1	S
0x4	1000BASE-LX				R					S	S	x1	S
0x5	1000BASE-KX			R			R			S		x1	S
0x6	1G-SGMII		R				R			S		x1	S
0x7	2.5GBASE-T	R						R		S	S	x1	S
0x8	2.5GBASE-X						R			S		x1	S
0x9	2.5GBASE-KX			R			R			S		x1	S
0xA	5GBASE-T	R						R		S	S	x1	S
0xB	5GBASE-KR			R			R			S		x1	S
0xC	10GBASE-T	R						R		S	S	x1	S
0xD	10G-SFI-DA					R				S	S	x1	S
0xE	10GBASE-SR				R					S	S	x1	S
0xF	10GBASE-LR				R					S	S	x1	S
0x10	10GBASE-KR			R			R		S	S		x1	S
0x11	10G-SFI-AOC/ACC				R					S	S	x1	S
0x12	10G-SFI-C2C						O			S		x1	S
0x78	2.5G SGMII											x1	S
0x79	100M-USXGMII		R									x1	S
0x7A	1G-USXGMII		R				R			S		x1	S
0x7B	2.5G-USXGMII		R				R			S		x1	S
0x7C	5G-USXGMII		R				R			S		x1	S

Table 3-217. Supported PHY Types [continued]

Index	Name	Auto-Neg			Media				FEC Type		Cage Type	PMD	POR
		C128	C137	C173	Optics/ Active	DAC	Back- plane	Twisted -pair	10G KR	No FEC	SFP	Width	
0x7D	10G-USXGMII		R				R			S		x1	S
Others	Reserved												

Some of the PHY Types above are valid only for the outermost link segment, while others can also apply for the internal link segments. To optimize the size of the topology netlist, there are two versions of the PHY Capabilities defined:

- **Extended PHY Capabilities Structure** — Used for the line-side of the outermost PHY. It is stored as a 128-bit wide bit-field and includes all supported PHY Types.
- **Basic PHY Capabilities Structure** — It is used for both line-side and host-side of the other PHYs in the PHY chain. It is stored as a 64-bit wide bit-field and only includes AU1 and backplane PHY Types.

All PHY nodes within the link topology capture these capability structures for both line and host-side of each port defined within a given port option.

3.8.3.1.1 Extended PHY Capabilities (128-bit Structure)

Table 3-218. 128-Bit Word Extended PHY Capabilities

Bit	Word[3]	Word[2]	Word[1]	Word[0]
0	Reserved	Reserved	Reserved	100BASE-TX
1	Reserved	Reserved	Reserved	100M-SGMII
2	Reserved	Reserved	Reserved	1000BASE-T
3	Reserved	Reserved	Reserved	1000BASE-SX
4	Reserved	Reserved	Reserved	1000BASE-LX
5	Reserved	Reserved	Reserved	1000BASE-KX
6	Reserved	Reserved	Reserved	1G-SGMII
7	Reserved	Reserved	Reserved	2.5GBASE-T
8	Reserved	Reserved	Reserved	2.5GBASE-X
9	Reserved	Reserved	Reserved	2.5GBASE-KX
10	Reserved	Reserved	Reserved	5GBASE-T
11	Reserved	Reserved	Reserved	5GBASE-KR
12	Reserved	Reserved	Reserved	10GBASE-T
13	Reserved	Reserved	Reserved	10G-SFI-DA
14	Reserved	Reserved	Reserved	10GBASE-SR
15	Reserved	Reserved	Reserved	10GBASE-LR
16	Reserved	Reserved	Reserved	10GBASE-KR/CR1
17	Reserved	Reserved	Reserved	10G-SFI-AOC/ACC
18	Reserved	Reserved	Reserved	10G-SFI-C2C
19	Reserved	Reserved	Reserved	
20	Reserved	Reserved	Reserved	
21	Reserved	Reserved	Reserved	
22	Reserved	Reserved	Reserved	
23	Reserved	Reserved	Reserved	
24	2.5G-SGMII	Reserved	Reserved	
25	100M-USXGMII	Reserved	Reserved	
26	1G-USXGMII	Reserved	Reserved	
27	2.5G-USXGMII	Reserved	Reserved	
28	5G-USXGMII	Reserved	Reserved	
29	10G-USXGMII	Reserved	Reserved	
30	Reserved	Reserved	Reserved	Reserved
31	Reserved	Reserved	Reserved	Reserved

3.8.3.1.2 Basic PHY Capabilities (64-bit Structure)

Table 3-219. 64-Bit Word Extended PHY Capabilities

Bit	Word[1]	Word[0]
0	Reserved	100M-SGMII
1	Reserved	1000BASE-KX
2	Reserved	1G-SGMII
3	Reserved	2500BASE-X
4	Reserved	2.5GBASE-KX
5	Reserved	5GBASE-KR
6	Reserved	10G-SFI-DA
7	Reserved	10GBASE-KR/CR1
8	Reserved	10G-SFI-C2C
9	Reserved	Reserved
10	Reserved	Reserved
11	Reserved	Reserved
12	Reserved	Reserved
13	Reserved	Reserved
14	Reserved	Reserved
15	Reserved	Reserved
16	Reserved	Reserved
17	Reserved	Reserved
18	Reserved	Reserved
19	Reserved	Reserved
20	Reserved	Reserved
21	Reserved	Reserved
22	Reserved	Reserved
23	Reserved	Reserved
24	2.5G-SGMII	Reserved
25	100M-USXGMII	Reserved
26	1G-USXGMII	Reserved
27	2.5G-USXGMII	Reserved
28	5G-USXGMII	Reserved
29	10G-USXGMII	Reserved
30	Reserved	Reserved
31	Reserved	Reserved

Note: Table 3-218 and Table 3-219 both include all PHY Types supported by Link Manager. Not all of them are applicable to a specific device. The link topology netlist only asserts capabilities shown as supported.

3.8.3.2 Port Options

The link topology can capture multiple sets of configuration information with different number of ports enabled and different port-to-lane mapping. See the Network Interface POR Configurations tables for the list of supported port options identified through their assigned Case#.

3.8.3.3 Adaptive NVM

The configuration required for any given port option defined within the link topology netlist can affect many E610 CSRs. This is implemented using the adaptive NVM method. The link topology netlist references two types of adaptive NVM feature tables:

- The Global super-feature table, affecting configurations that should be modified according to the combination of the valid ports and their corresponding maximal speed
- Per-port / Per-function super-features tables, affecting configurations relevant for the specific port type and speed.

The configuration for the global super-feature is applied based on the Global Super Configuration ID taken from the currently active port option of the innermost PHY (i.e. the E610 PHY node).

The configuration for the per-port super-features are applied based on the Per Port Feature ID taken from the innermost PHY and the and Per Port Configuration ID AND-ed along the PHY chain associated with a given port.

The PF to Port Mapping Configuration ID is also taken from the innermost PHY.

The configuration for the per-function super-features are applied based on the Per Function Feature ID taken from the innermost PHY and the Per Function Configuration ID from the outermost PHY.

For the list of all valid feature IDs and corresponding configuration IDs. refer to [Section 3.8.5, "Link Topology Adaptive NVM Features"](#).

3.8.3.4 Link Topology Netlist

The topology netlist is a description of all components of a specific board associated with the Ethernet link, a description of the way to control those components, and the connectivity between them. The topology netlist includes an entry for each component within (node), with a description of its device type, its interfaces, capabilities, associated configuration, and any information needed to setup this device in order to bring up the Ethernet link.

Each topology node is composed of several sections. Two sections are common for all topology nodes, the Node Header Section that includes the basic description of the node and pointers to the other sections and the Node I/O Section that includes pointers and descriptions of the node I/Os that connect the node toward the host and are reflected the topology netlist.

Some topology devices do not need any additional description besides the header section and the node I/O section, while others include additional sections to reflect topology related properties that are specific to the node.

Some nodes allow additional node specific for run-time information to be stored in a provided scratch section, which is not written to disk and is not part of the stored netlist used by the scripts to store scratch data.

The following basic topology devices contain only the node header and node I/O section, while some of the node can also contain the scratch area (node scratch area):

- GPIO Controller Node
- MUX Controller Node
- LED Controller Node
- Cage Node

The Node Header Section contains information on the following:

- Node type, node handle, node address, node part number, node options
- I/O section pointer
- Port options section pointer — Relevant for PHY, LED, and Temperature sensor nodes and includes Port option count and the pointer to the Port options section.
- Line-side analog Port options section pointer — Relevant for PHY nodes and includes the pointer to the analog Port options section.
- Host-side analog Port options section pointer — Relevant for PHY nodes and includes the pointer to the analog Port options section.

The Node I/O Section includes a list of interfaces that are connected to a specific node. The link topology netlist is built as a tree where the root represents components integrated within the device. Therefore, it is sufficient that each topology node would list only the interfaces that connects between the node and other nodes that are up in the topology tree. These I/Os could be either I/Os that are controlled by this node or I/Os that are used to control this node. Within the node I/O section, each such interface contains the following information:

- Driving node handle — This includes a node handle, used to point to the topology device that this interface is connected to and the I/O number at that topology device.
- I/O type and driving interface — Contains information about the type of the interface (ex. PMD, I²C, MDIO, GPIO, LED), the function of the interface (ex. when the type is SDP - what is the functionality of that SDP - INT_N / PRESET_N / TX_DISABLE etc.).
- Other properties of the I/Os like Polarity, Strapped, Strap value, Interrupt and Speed. Those properties might be used with the proper I/O type.

The PHY Node includes several sections in addition to the above. It adds two sections of Node PMD Analog Section and Node PMD Analog Misc Section, one each for the host-side and one each for the line-side. PHY nodes may capture configuration information for various number of ports enabled on them also referred to as Port Options. To easily locate these Port Options the PHY node includes a Node Port Option Pointer Section with pointers to each of the port options. For each of the captured port options the PHY Node includes a Node Port Option Header Section and for each of the ports defined within the port option a Node PHY Capabilities Section.

The Node PMD Analog Section contains the following information:

- Coefficients for 1G/2.5G/5G, 10G C2C and C2M AUI modes.
- Link establishment state machine timeouts for 1G/2.5G/5G, 10G AUI modes.
- Link establishment state machine timeouts for auto-negotiated modes.
- Per lane transmit and receive polarity.

The Node Port Option Header Section includes the following information:

- Super Configuration ID — A pointer used to select the corresponding EPL configuration.
- Minimum SKU — The minimum SKU that this port option can work with.
- PMD count — The number of PMDs within this port option.
- A pointer to the Node PHY capabilities section per PMD.

The Node PHY Capabilities Section includes the following information:

- PHY Capabilities Structures for the line and host-side of interfaces of the node.
- Link options (EEE enable, FEC abilities advertise and request, PCS enable, Low power mode enable, LPLU enable, Rx and Tx pause, Auto-neg options (cl. 37,73), LESM enable, Host and line lane reversal, outermost indication).
- PMD width and lane allocation for host-side and line-side.

LED Nodes and Temperature Sensor Nodes include a Node Parent Section which allows them to be associated with a given PHY Node and allows selection of the appropriate configuration based on the active port option of the PHY Node. The port option selected for the parent device will be applied also for the current topology device.

The LED Node includes the basic node sections, the node parent section and a Node LED Configuration Section for each port option of its parent node, with the following information:

- LED link affinity — Up to 24 bits (lowest 24 bits) are used to associate the Ethernet link with the LED. When the bit is set, the relevant Ethernet link will affect the LED. The highest two bits, will be used to select the logic operation between the ports (AND or OR).
- LED flag.
- LED color and condition.

The Temperature Sensor Nodes includes the basic node sections, the node parent section and a Node Thermal Configuration Section for each port option, with the following information:

- Temperature sensor link Affinity (Low / High) — Up to 32 bits are used to associate the Ethernet link with the temperature sensor. Each link should be associated with one temperature sensor.
- Temperature alarm threshold — The threshold temperature for this sensor.

Table 3-220 summarize the link topology node structure, for different nodes:

Table 3-220. Link Topology Node Structure

Section Name	Description	Applicable to:	Per Node Section Count
Header Section	This section contains basic information for all node types, such as node type, section length, address, and pointers to the other node sections.	All node types	One section. Fixed size.
I/O Section	This section contains information about the connectivity of the node within the topology netlist.	All node types	One section. Size is linear with the number of I/Os.
LED Configuration Section	This section contains information about the configuration of the LED, such as polarity, port affinity, and color.	LED nodes	One section. Size is linear with the number of port options of the parent PHY node.
Thermal Configuration Section	This section contains information dealing with the configuration of the thermal sensor, such as port affinity and alarm threshold.	Temperature Sensor nodes	One section. Size is linear with the number of port options of the parent PHY node.

Table 3-220. Link Topology Node Structure [continued]

Section Name	Description	Applicable to:	Per Node Section Count
PMD Analog Section	This section contains the configuration of the PHY lanes analog side.	PHY nodes	Two sections per PHY node, one each for the host and line-side. Size is linear with the number of lanes.
PMD Analog Misc Section	This section contains PHY specific configuration of the PHY lanes analog side.	PHY nodes	Two sections per PHY node, for the host and line-side. Size is linear with the number of lanes.
Port Option Pointer Section	This section contains the pointers for all PMD options within a single PHY node.	PHY nodes	One section per PHY node. Size is linear with the number of port options.
Port Option Header Section	This section contains the number of PMDs per port option and pointers to Node PHY Capabilities Section.	PHY nodes	Each PHY node has sections for each port option. Size is linear with the number of PMDs within the port option.
Node PHY Capabilities Section	This section contains the configuration of the PHY in a specific port option.	PHY nodes	Each port option has sections for each PMD. Size is fixed.

3.8.4 Link Topology Use Case Examples

This section provides a high-level visual representation of the resulting link topology netlist, for a specific use case.

3.8.4.1 Single Port SFP NIC Use Case

Figure 3-51 describes typical NIC use case highlighting the relevant link topology and showing resulting link topology structures.

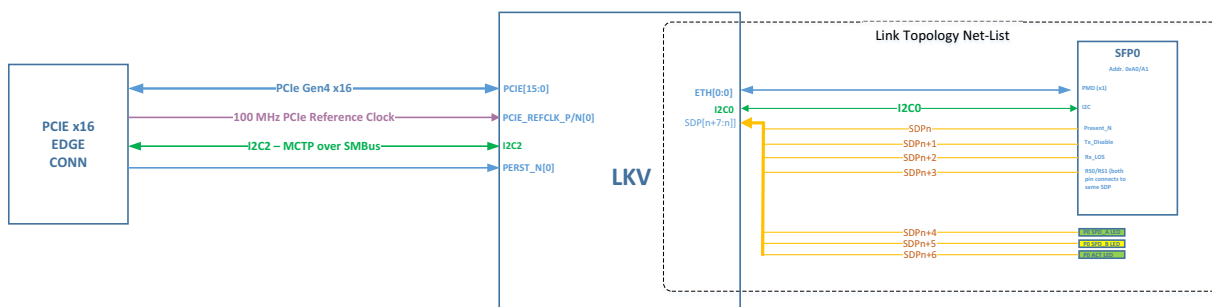


Figure 3-51. Single Port SFP NIC Topology Use Case

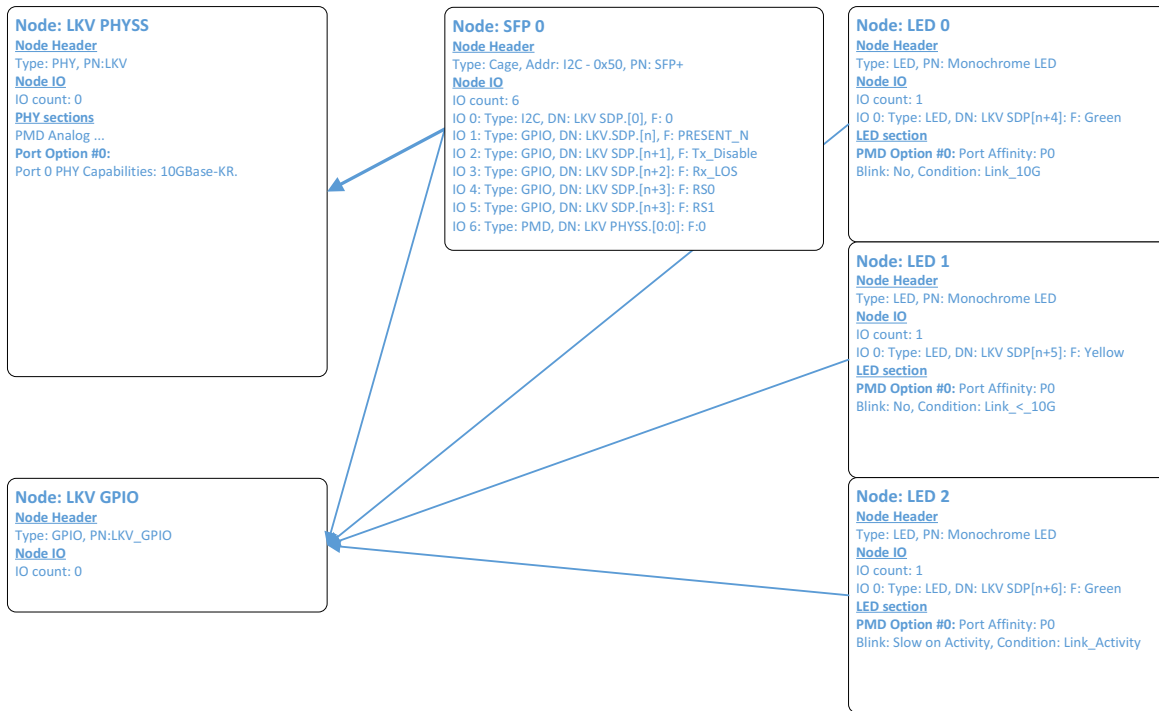


Figure 3-52. Single Port SFP NIC Topology Data Structures

3.8.4.2 Dual Port SFP NIC Use Case

Figure 3-53 describes typical NIC use case highlighting the relevant link topology and showing resulting link topology structures.

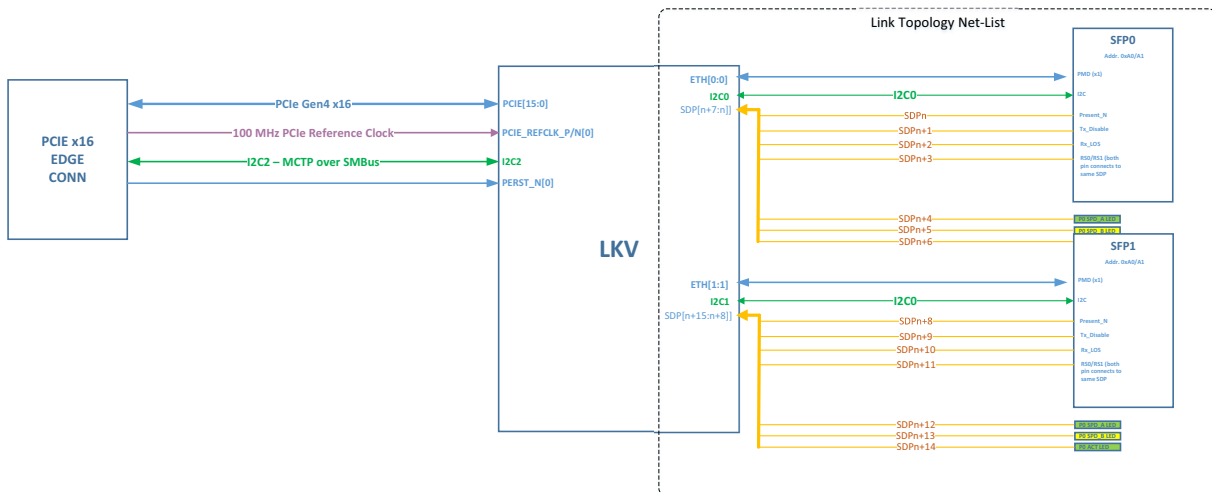


Figure 3-53. Dual Port SFP NIC Topology Use Case

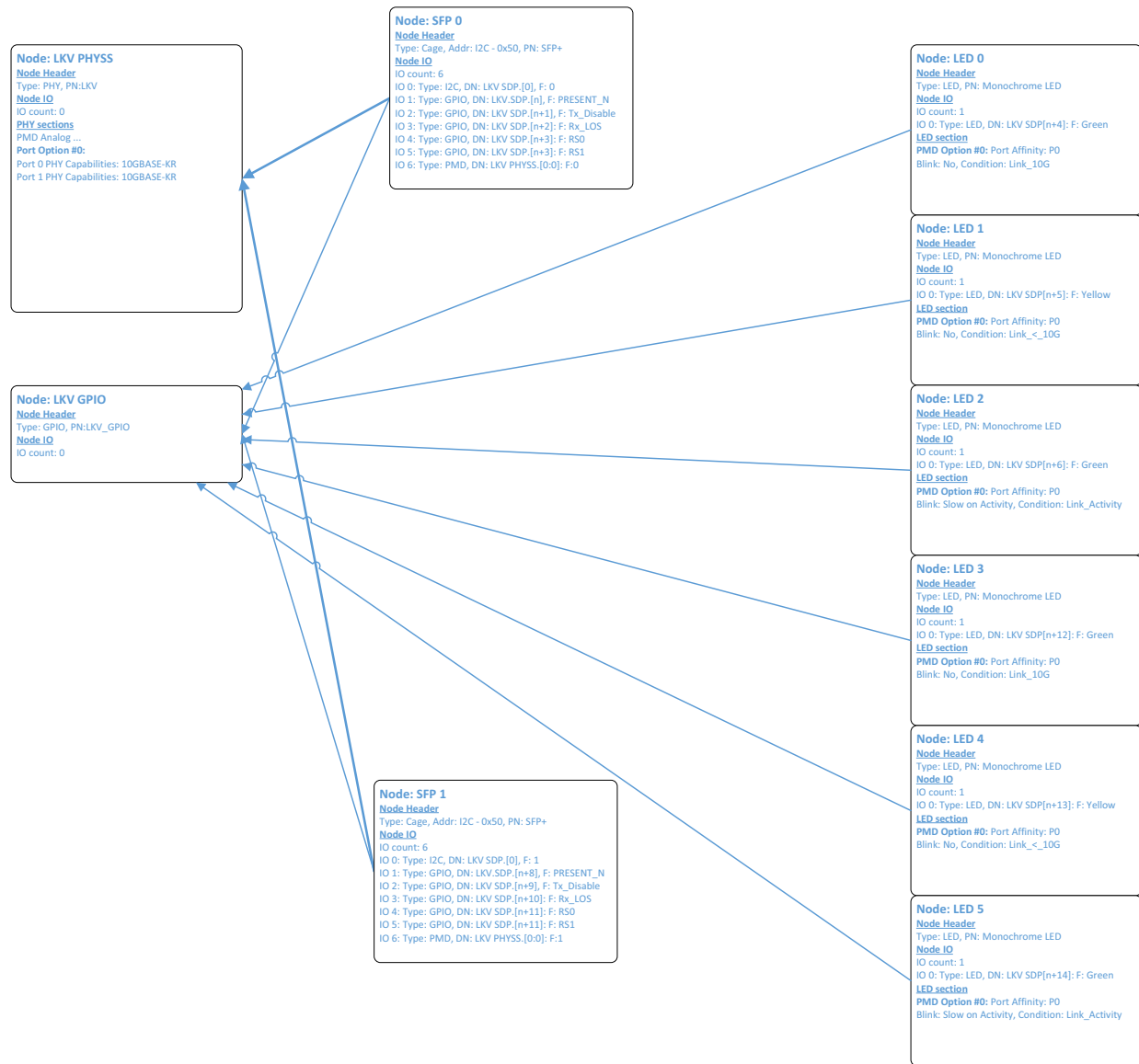


Figure 3-54. Dual Port SFP NIC Topology Data Structures

3.8.5 Link Topology Adaptive NVM Features

3.8.5.1 Overview

A port option can be defined for each valid combination of E610 ports. To configure the device accordingly, a set of corresponding global super-configurations and set of per-port and per-function features and configurations are defined. This section details the relevant features, super features, and their possible configuration options. The configurations defined for these features and super features include all that is required for supporting all E610 configurations.

3.8.5.2 Features

3.8.5.2.1 Per-Port Features

Table 3-221. Port Enable[0:1]

Order	Config ID	Description
1.0	0x0001	Enable
2.0	0x0002	Disable

Table 3-222. WoL -Port [0:1] (Auto Generated)

Order	Config ID	Description
1.0	0x0001	Disable
2.0	0x0002	APM Enable

3.8.5.2.2 Per-Function Features

Table 3-223. PF Enable[0:1]

Order	Config ID	Description
1.0	0x0001	Enable
2.0	0x0002	Disable

Table 3-224. Device ID - PF[0:1]

Order	Config ID	Description
0.0	0x0000	E610 - Disable
1.0	0x0001	E610 - Default Device ID
2.0	0x0002	E610 - Default VF Device ID
3.0	0x0003	E610 - Backplane Device ID
5.0	0x0005	E610 - SFP Device ID
9.0	0x0009	E610 - 1000BASE-T Device ID
10.0	0x000A	E610 - SGMII Device ID

3.8.5.2.3 Global Features

Table 3-225. Number of E610 Ports

Order	Config ID	Description
1.0	0x0001	1 Port
2.0	0x0002	2 Ports

Table 3-226. Number of PFs

Order	Config ID	Description
1.0	0x0001	1 PF NIC Mode
1.1	0x0101	1 PF FPGA
2.0	0x0002	2 PF NIC Mode
2.1	0x0102	2 PF FPGA

Table 3-227. Manageability Mode (Feature ID is 0x0009)

Order	Config ID	Description
1.0	0x0001	NC-SI
2.0	0x0002	No MNG
3.0	0x0003	Full MCTP w/ PLDM
4.0	0x0004	NCSI w/ PLDM over MCTP control
5.0	0x0005	Full MCTP w/ PLDM, with non-default values
6.0	0x0006	Full MCTP w/o PLDM

Table 3-228. PF-to MAC-Mapping (Feature ID is 0x000A)

Order	Config ID	Description
1.0	0x0001	Standard 2 Port Mapping (0,1) -> (0,1)
2.0	0x0002	Inverted 2 Port Mapping (0,1) -> (1,0)

Table 3-229. Ports DCB-PM Configuration

Order	Config ID	Description
1.0	0x0001	1 Port: 1x1G (Port 0)
2.0	0x0002	2 Port: 2x1G (Port 0, 1)

3.8.5.2.4 Super Features

Table 3-230. Per-Port Super Features 0:1 (Super Features IDs are 0xF0E0 - 0xF0E7)

Order	Config ID	Description
0,0	0x0000	Disable
1.0	0x0001	Enable, WoL Disable
2.0	0x0002	Enable, WoL enabled

Table 3-231. Per-Function Super Features (Super Features IDs are 0xF0F0 - 0xF0F7)

Order	Config ID	Description
0,0	0x0000	E610 - Disable
1.0	0x0001	E610 - Silicon Default
2.0	0x0002	E610 - Default VF
3.0	0x0003	E610 - Backplane
5.0	0x0005	E610 - SCP SFP/SFP+
9.0	0x0009	E610 - MCP 1GBASE-T
10.0	0x000A	E610 - SGMII

3.8.5.2.5 Global Super Features

Table 3-232. Number of E610 Ports

Order	Super Config ID	Description
1.0	0x0001	1 Port: 1x1G (Port 0)
2.0	0x0002	2 Port: 2x1G (Port 0, 1)

3.8.6 Link Topology Netlist

3.8.6.1 Overview

This section defines the structure of the topology netlist. The structure begins with a header followed by DWord-aligned blocks describing the netlist nodes in the following order: PHY Nodes, GPIO Controller Nodes, MUX Controller Nodes, LED Controller Nodes, LED Nodes, Temperature Sensor Nodes, ID EEPROM Node, Cage Nodes, and Mezzanine Connector Nodes.

For each of the nodes the netlist contains a block with separate sections dedicated to different features of the node. Presence of certain sections depends on the node type.

3.8.6.2 Node Sections

3.8.6.2.1 Topology Netlist Header

Defines the length of the Topology Netlist Section and the number of nodes it holds.

Word Offset	Description	Section Reference
0x0000	Module Length	3.8.6.2.1.1
0x0001	Node Count	3.8.6.2.1.2
0x0002	Netlist Map Version and CRC	3.8.6.2.1.3
0x0003	Netlist Version	3.8.6.2.1.4
0x0004 + n*2	Node Handle — One for each node.	3.8.6.2.1.5
0x0005 + n*2	Node Block Offset — One for each node.	3.8.6.2.1.6
0x0004 + Node Count*2	Netlist Identifier Section This is a single block following the n Node Handle and Node Block Offset pointers.	3.8.6.2.2

3.8.6.2.1.1 Module Length (0x0000)

Bits	Field Name	Value	Description
15:0	Module Length		Module length in double words. Does not include the <i>Module Length</i> field itself and the <i>Node Count</i> field.

3.8.6.2.1.2 Node Count (0x0001)

Bits	Field Name	Value	Description
15:12	Netlist Version	1	The version of the Netlist. This field changes when the content of a tracked netlist changes. In other words, the values of some of the fields in the netlist change. It reflects the short hash of the GIT commit corresponding with the last change. Netlist Version [3:0]
11:10	Reserved		Reserved.
9:0	Node Count		The number of nodes captured in the topology netlist.

3.8.6.2.1.3 Netlist Map Version and CRC (0x0002)

Bits	Field Name	Value	Description
15:8	Netlist Map Version	0x20	The version of the netlist field definition. This field changes every time there is a change in the definition of the netlist, like fields, structures, or nodes are added, removed, or modified.
7:0	CRC8		CRC-8-CCITT: Start Section -> Word offset = 0x0004 End Section -> Word offset = Module Length

3.8.6.2.1.4 Netlist Version (0x0003)

Bits	Field Name	Value	Description
15:0	Netlist Version	0	The version of the Netlist. This field changes when the content of a tracked netlist changes. In other words, the values of some of the fields in the netlist change. It reflects the short hash of the GIT commit corresponding with the last change. Netlist Version [19:4]

3.8.6.2.1.5 Node Handle (0x0004 + n*2)

Bits	Field Name	Value	Description
15:10	Reserved		Reserved.
9:0	Node Handle		Unique identifier of the node in the topology. Matches the <i>Node Handle</i> field inside the corresponding Node Header Section.

3.8.6.2.1.6 Node Block Offset (0x0005 + n*2)

Bits	Field Name	Value	Description
15:0	Node Block Offset		The word offset of the node section relative to the beginning of the module.

3.8.6.2.2 Netlist Identifier Section

This section defines the structure of the Netlist Identifier.

Word Offset	Description	Section Reference
0x0000	Module ID 0x0001 This is new. See bottom for list of module IDs.	3.8.6.2.2.1
0x0001	Module Length For Netlist Identifier block.	3.8.6.2.2.2
0x0002	Base Release Version: Major Low	3.8.6.2.2.3
0x0003	Base Release Version: Major High	3.8.6.2.2.4
0x0004	Base Release Version: Minor Low	3.8.6.2.2.5
0x0005	Base Release Version: Minor High	3.8.6.2.2.6
0x0006	Base Release Version: Type Low	3.8.6.2.2.7
0x0007	Base Release Version: Type High	3.8.6.2.2.8
0x0008	Base Release Version: Revision Low	3.8.6.2.2.9
0x0009	Base Release Version: Revision High	3.8.6.2.2.10
0x000A-0x0019	Netlist Binary Hash (256-bit Hash)	3.8.6.2.2.11
0x001A	Netlist Origin Flags Low	3.8.6.2.2.12
0x001B	Netlist Origin Flags High	3.8.6.2.2.13
0x001C	Netlist Modification Date: Year	3.8.6.2.2.14
0x001D	Netlist Modification Date: Day/Month	3.8.6.2.2.15
0x001E	Netlist Modification Date: Time	3.8.6.2.2.16
0x001F	ETT Version Major	3.8.6.2.2.17
0x0020	ETT Version Minor	3.8.6.2.2.18
0x0021	ETT Version Revision	3.8.6.2.2.19
0x0022	ETT Version Patch	3.8.6.2.2.20
0x0023-0x002C	Full GIT Netlist SHA-1 Hash	3.8.6.2.2.21
0x002D	Customer IANA Low	3.8.6.2.2.22
0x002E	Customer IANA High	3.8.6.2.2.23
0x002F	Customer Netlist Version	3.8.6.2.2.24

Note: The section is not accessible through the GETNODEATTR PHY configuration action.

3.8.6.2.2.1 Module ID (0x0000)

Bits	Field Name	Value	Description
15:0	Module ID	0x0000	Module ID. 0x0000 = Reserved 0x0001 = Netlist Identifier Section 0x0002-0xFFFF = Reserved

3.8.6.2.2.2 Module Length (0x0001)

Bits	Field Name	Value	Description
15:0	Module Length	0x0000	Module length in double words. Does not include the <i>Module Length</i> field.

3.8.6.2.2.3 Base Release Version: Major Low (0x0002)

Bits	Field Name	Value	Description
15:0	Base Release Version: Major Low	0x0000	Low word of the Base Release Version: Major value. Stored in binary coded decimal (BCD). Similar to NVM version.

3.8.6.2.2.4 Base Release Version: Major High (0x0003)

Bits	Field Name	Value	Description
15:0	Base Release Version: Major High	0x0000	High word of the Base Release Version: Major value. Stored in binary coded decimal (BCD). Similar to NVM version.

3.8.6.2.2.5 Base Release Version: Minor Low (0x0004)

Bits	Field Name	Value	Description
15:0	Base Release Version: Minor Low	0x0000	Low word of the Base Release Version: Minor value. Stored in binary coded decimal (BCD). Similar to NVM version.

3.8.6.2.2.6 Base Release Version: Minor High (0x0005)

Bits	Field Name	Value	Description
15:0	Base Release Version: Minor High	0x0000	High word of the Base Release Version: Minor value. Stored in binary coded decimal (BCD). Similar to NVM version.

3.8.6.2.2.7 Base Release Version: Type Low (0x0006)

Bits	Field Name	Value	Description
15:0	Base Release Version: Type Low	0x0000	Low word of the Base Release Version: Type value. Each unique netlist from Intel has a different Type value. Stored in binary coded decimal (BCD). Similar to NVM version.

3.8.6.2.2.8 Base Release Version: Type High (0x0007)

Bits	Field Name	Value	Description
15:0	Base Release Version: Type High	0x0000	High word of the Base Release Version: Type value. Each unique netlist from Intel has a different Type value. Stored in binary coded decimal (BCD). Similar to NVM version.

3.8.6.2.2.9 Base Release Version: Revision Low (0x0008)

Bits	Field Name	Value	Description
15:0	Base Release Version: Revision Low	0x0000	Low word of the Base Release Version: Revision value. This is updated by ETT on any change to the template.

3.8.6.2.2.10 Base Release Version: Revision High (0x0009)

Bits	Field Name	Value	Description
15:0	Base Release Version: Revision High	0x0000	High word of the Base Release Version: Revision value. This is updated by ETT on any change to the template.

3.8.6.2.2.11 Netlist Binary Hash (0x000A-0x0019)

Bits	Field Name	Value	Description
15:0	Hash Value of Netlist	0x0000	Hash of Netlist complied binary. Inserted by ETT as the last step of netlist creation. This should be inserted and updated on any netlist change by Intel or customer. It should be the SHA3 256-bit hash of the netlist where the Hash Field is set all zeros. The binary hash should be calculated assuming that both the binary hash field and the CRC8 field in the netlist header is zero. And once the binary hash is injected, the CRC8 for the netlist header should be calculated as the final step for the netlist binary compilation.

3.8.6.2.2.12 Netlist Origin Flags Low (0x001A)

Bits	Field Name	Value	Description
15:3	Reserved	0x0000	Reserved.
2	Intel Created Custom		0b = Not custom. 1b = Intel-created net for use with a customer netlist by OEM.
1	Intel Factory Created		0b = Not created by Intel factory. 1b = Intel factory-created net for use as a reference netlist.
0	Netlist Origin	0	0b= Intel-created and unmodified by customer. 1b= Customer modified netlist. ETT must set this bit on any customer change of the netlist.

3.8.6.2.2.13 Netlist Origin Flags: High (0x001B)

Bits	Field Name	Value	Description
15:0	Reserved	0x0000	Reserved.

3.8.6.2.2.14 Netlist Modification Date: Year (0x001C)

Bits	Field Name	Value	Description
15:0	Netlist Modification Date: Year	0x0000	Year of the last netlist modification in binary coded decimal (BCD). Set by ETT on any Intel or customer modification of the netlist. Example: 0x2019 = 2019.

3.8.6.2.2.15 Netlist Modification Date: Day/Month (0x001D)

Bits	Field Name	Value	Description
15:8	Netlist Modification Date: Day	0x00	Day of the month of last netlist modification in binary coded decimal (BCD). Set by ETT on any Intel or customer modification of the netlist. Example: 0x20 = 20th day of the month.
7:0	Netlist Modification Date: Month	0x00	Numerical representation of the Month of last netlist modification in binary coded decimal (BCD). Set by ETT on any Intel or customer modification of the netlist. Example: 0x11 = the 11th months, which is November.

3.8.6.2.2.16 Netlist Modification Date: Time (0x001E)

Bits	Field Name	Value	Description
15:0	Netlist Modification Date: Time	0x0000	Time of the last netlist modification in binary coded decimal (BCD). Set by ETT on any Intel or customer modification of the netlist. Example: 0x1300 is 1:00 pm.

3.8.6.2.2.17 ETT Version Used for Netlist Compile: Major (0x001F)

Bits	Field Name	Value	Description
15:0	ETT Major Version	0x0000	Major version of ETT used to compile the most recent version of the netlist. This should be inserted and updated on any netlist change by Intel or customer.

3.8.6.2.2.18 ETT Version Used for Netlist Compile: Minor (0x0020)

Bits	Field Name	Value	Description
15:0	ETT Minor Version	0x0000	Minor version of ETT used to compile the most recent version of the netlist. This should be inserted and updated on any netlist change by Intel or customer.

3.8.6.2.2.19 ETT Version Used for Netlist Compile: Revision (0x0021)

Bits	Field Name	Value	Description
15:0	ETT Revision Version	0x0000	Revision version of ETT used to compile the most recent version of the netlist. This should be inserted and updated on any netlist change by Intel or customer.

3.8.6.2.2.20 ETT Version Used for Netlist Compile: Patch (0x0022)

Bits	Field Name	Value	Description
15:0	ETT Patch Version	0x0000	Patch version of ETT used to compile the most recent version of the netlist. This should be inserted and updated on any netlist change by Intel or customer.

3.8.6.2.2.21 Full GIT Netlist SHA-1 Hash (0x0023-0x002C)

Bits	Field Name	Value	Description
15:0	Full Git Hash Value	0x0000	Full GIT commit hash of netlist. Should be set by ETT when committing changes to GIT.

3.8.6.2.2.22 Customer IANA Low (0x002D)

Bits	Field Name	Value	Description
15:0	Customer IANA Low	0x0157	Customer IANA low. Intel to set Intel IANA 0x0157. Customer to fill in their IANA.

3.8.6.2.2.23 Customer IANA High (0x002E)

Bits	Field Name	Value	Description
15:0	Customer IANA High	0x0000	Customer IANA high. Intel to set Intel IANA 0x0000. Customer to fill in their IANA.

3.8.6.2.2.24 Customer Netlist Version (0x002F)

Bits	Field Name	Value	Description
15:0	Customer Netlist Version	0x0000	Customer version that customer can set in customer ETT. Template images from Intel would set this to 0.

3.8.6.2.3 Node Header Section

This section defines the structure of the node header detailing the node type, node block length, Node Handle and Node Part Number. Based on node type where applicable the number of I/Os, Port Options, Module Qualification options, firmware download options, and conflict resolution options are also defined.

Word Offset	Description	Section Reference
0x0000	Node Type and Node Section Length	3.8.6.2.3.1
0x0001	Node Handle	3.8.6.2.3.2
0x0002	Node Address	3.8.6.2.3.3
0x0003	Node Part Number and Node Options	3.8.6.2.3.4
0x0004	Node I/O Section Pointer	3.8.6.2.3.5
0x0005	Node Port Options Section Pointer	3.8.6.2.3.6
0x0006	Node PMD Line Analog Section Pointer	3.8.6.2.3.7
0x0007	Node PMD Host Analog Section Pointer	3.8.6.2.3.8

The section is accessible through the GETNODEATTR DNL action starting at the DWord-aligned Attribute ID offset 0x0000. The Node Address within Attribute ID 0x0001 can also be modified through the SETNODEATTR DNL action.

3.8.6.2.3.1 Node Type and Section Length (0x0000)

Bits	Field Name	Value	Description
15:12	Node Type		Defines the node type. Valid values are: 0x00 = PHY device 0x01 = GPIO Controller (Port Expander) 0x02 = MUX Controller (Bus Expander) 0x03 = LED Controller (LED Driver) 0x04 = LED 0x05 = Temperature Sensor 0x06 = Cage (SFP, SFP28, D SFP, SFP DD) 0x09 = Clock Controller (DPLL/Synthesizer/Divider) 0x10 = Clock MUX 0x11 = GPS device All other values are reserved.
11:0	Node Section Length		Node section length in double words. Does not include the first double word, the <i>Node Section Length</i> field itself.

3.8.6.2.3.2 Node Handle (0x0001)

Bits	Field Name	Value	Description
15:10	Reserved		Reserved.
9:0	Node Handle		Unique identifier of the node in the topology. Values must be assigned sequentially from 0 to Node Count - 1. For LOM: Bit Description 9 Board Type — Identifies if the node is on the motherboard or a mezzanine card. 0b = LOM 8:0 Unique node identifier within the context of the motherboard. For Mezzanine Cards: Bit Description 9 Board Type — Identifies if the node is on the motherboard or a mezzanine card. 1b = Mezzanine Card 8:6 Mezzanine card number. Populated by firmware during the mezzanine discovery flow. The value in the mezzanine topology is always 3'b0. 5:0 Unique node identifier within the context of the mezzanine card.

3.8.6.2.3.3 Node Address (0x0002)

Bits	Field Name	Value	Description
15:14	Module Max Power Hi		Module Max Power [5:4] - The upper two bits of <i>Module Max Power</i>. The field represents the maximum allowed module power with a 0.5 W resolution. Valid for Cage nodes only and reserved for all other node types. Note: The value of <i>Module Max Power</i> must be greater than or equal to 2 (1.0 W) for SFP cage nodes.
13:12	Bus Address Type		Defines whether the bus address is absolute or relative to the mezzanine card (CEI) slot. 00b = Reserved 01b = Absolute address. The uniqueness of the address is ensured by the topology. 10b = Relative address. The address must be updated on board initialization to avoid conflicts. 11b = Reserved. Not applicable to LED nodes.
11:10	Bus Type		Defines the bus type. Valid options are: 00b = Internal Buses (for example, IOSF-SB/AXI) 01b = I²C 10b = MDIO 11b = Reserved Not applicable to LED nodes.
9:0	Bus Address		For all node types other than LED: The bus address that the node can be accessed at. The range of valid addresses depends on the bus type. For internal buses (for example, IOSF-SB/AXI), the assignment is as follows (mapping to the actual bus and address is done by the DNL scripts): 0x000 = Integrate PHY - E610 PHY 0x003 = Integrated GPIO/I²C/MDIO Controller - E610 I/O Widget All other values are reserved. For I²C, the field represents the 7-bit or 10-bit address of the I²C attached node: 0x000-0x07F = For 7-bit addressable I²C devices. The 3 high MSB should be set to 0x0. 0x000-0x3FF = For 10-bit addressable I²C devices. For MDIO, the field represents the 5-bit Port Address: 0x000-0x01F = 5-bit MDIO Port Address. 0x020-0x3FF = Reserved.

3.8.6.2.3.4 Node Part Number and Node Options (0x0003)

Bits	Field Name	Value	Description
15:8	Node Part Number		The part number assigned to each of the POR devices supported by the topology and link management. The defined part numbers are: 0x11 = SFP+ Cage (Node type - Cage) 0x12 = SFP28 Cage (Node type - Cage) 0x15 = Monochrome LED (Node type - LED) 0x16 = RGB LED (Node type - LED) 0x20 = PCA9545A 0x21 = PCA9575 (Node type - GPIO controller) 0x22 = PCA9685 (Node type - LED controller) 0x23 = PCA9552 (Node type - LED controller) 0x24 = ZL30632/ZL80032 (Node type - Clock Controller) 0x25 = SI5383 (Node type - Clock Controller) 0x2F = MV-CUX3610 (Node type - PHY, Temperature Sensor, Clock Controller) 0x31 = C827 (Node type - PHY, Temperature Sensor, Clock Controller) 0x32 = X557-AT4 (Node type - PHY, Temperature Sensor) 0x33 = 88E1543 (Node type - PHY, Temperature Sensor) 0x34 = 88E1512 (Node type - PHY, Temperature Sensor, Clock Controller) 0x35 = 88E1514 (Node type - PHY, Temperature Sensor) 0x3B = X557-AT2 (Node type - PHY, Temperature Sensor) 0x44 = SFP+/SFP28 Module (Node type - Temperature Sensor) 0x47 = Generic Clock MUX 0x48 = Generic GPS 0x49 = E610 (Node type - PHY, Temperature Sensor, Clock Controller) 0x4A = E610 (Node type - GPIO Controller) All other values are reserved.
7:4	Module Max Power Lo		Module Max Power [3:0] - The lower four bits of <i>Module Max Power</i>. The field represents the maximum allowed module power with a 0.5 W resolution. Valid for Cage nodes only and reserved for all other node types. Note: The value of <i>Module Max Power</i> must be greater than or equal to 2 (1.0 W) for SFP cage nodes.
3	Module Qualification Enable		Module qualification enable. 0b = Pluggable module qualification is disabled for the cage. 1b = Pluggable module qualification is enabled for the cage. Valid for Cage nodes only. Reserved for all other node types.
2	Innermost PHY		Defines whether the PHY is the innermost PHY: 0b = The PHY has host-side PMD connections to another PHY node. 1b = The PHY has host-side connections to the MAC. Valid for PHY nodes only. Reserved for all other node types.
1	Reserved		Reserved.
0	FW Download Enable		Firmware download enable. 0b = The PHY loads its firmware from a dedicated EEPROM/Flash after reset, or does not require firmware. 1b = The PHY firmware should be downloaded from NVM at initialization. Valid for PHY nodes only. Reserved for all other node types.

3.8.6.2.3.5 Node I/O Section Pointer (0x0004)

Bits	Field Name	Value	Description
15:12	I/O Count		The number of host-side I/O connections of the node. This defines the length of the Node I/O Section.
11:0	Node I/O Section Offset		The word offset of the Node I/O Section relative to the beginning of the node section.

3.8.6.2.3.6 Node Port Options Section Pointer (0x0005)

Bits	Field Name	Value	Description
15:12	Port Option Count		Defines the number of Port Options supported by the node. This defines the number Node PHY Port Options. The LED node and Thermal sensor LED inherit the number of Port Options from their Parent PHY node. Not applicable to nodes of type other than PHY. For those nodes this field is reserved and must be 0x0.
11:0	Node Port Options Section Offset		For PHY nodes: The word offset of the Node Port Options Section relative to the beginning of the node section. For LED nodes: The word offset of the first Node LED Configuration Section relative to the beginning of the node section. For Temperature Sensor nodes: The word offset of the Node Thermal Configuration Section relative to the beginning of the node section. Not applicable to any other node types.

3.8.6.2.3.7 Node PMD Line Analog Section Pointer (0x0006)

Bits	Field Name	Value	Description
15:12	Line Lane Count		The number of line-side lanes on the PHY node.
11:0	Node PMD Line Analog Section Offset		The word offset of the Node PMD Line Analog Section relative to the beginning of the node section.

3.8.6.2.3.8 Node PMD Host Analog Section Pointer (0x0007)

Bits	Field Name	Value	Description
15:12	Host Lane Count		The number of host-side lanes on the PHY node.
11:0	Node PMD Host Analog Section Offset		The word offset of the Node PMD Host Analog Section relative to the beginning of the node section.

3.8.6.2.4 Node I/O Section

This section defines the structure of the section detailing the I/O connections of a topology node.

Word Offset	Description	Section Reference
0x0000 + n*2	Driving Node Handle	3.8.6.2.4.1
0x0001 + n*2	I/O Type and Driving Interface	3.8.6.2.4.2

Note: In this context, “n” refers to the index of the node I/O connections and ranges from 0 to I/O Count - 1.

The section is accessible through the GETNODEATTR DNL action starting at the DWord-aligned Attribute ID offset 0x2000.

3.8.6.2.4.1 Driving Node Handle (0x0000 + n*2)

Bits	Field Name	Value	Description
15:10	Driving I/O Number		The number of the interface with the specified <i>I/O Type</i> on the device pointed to by <i>Driving Node Handle</i> to which the connection leads.
9:0	Driving Node Handle		Unique identifier of the node in the topology to which the I/O is connected.

3.8.6.2.4.2 I/O Type and Driving Interface (0x0001 + n*2)

Bits	Field Name	Value	Description
15	Driven		Defines if the I/O is strapped to a certain value or is driven by another node in the topology. 0b = The I/O is strapped to a value defined by the <i>Value</i> field. 1b = The I/O is driven by the device identified by the <i>Driving Node Handle</i> . Only applicable to I/Os with <i>I/O Type</i> of GPIO and MDIO Address.
14	Value		For driven I/Os, defines the default value. For strapped I/Os, defines the static value of the I/O to which it is configured. Only applicable to I/Os with <i>I/O Type</i> of GPIO and MDIO Address. Note: For driven I/Os that use inverted I/O function (like, Driven = 1 and Polarity = 0), the signal is driven to the inverse of this field.
13	Polarity		Polarity configuration for the I/O. 0b = The signal is active low. 1b = The signal is active high. Only applicable to I/Os with <i>I/O Type</i> of GPIO.
12:11	Interrupt		Interrupt configuration for the I/O. 00b = No interrupt. 01b = Interrupt on rising edge. 10b = Interrupt on falling edge. 11b = Interrupt on change (rising or falling edge). Only applicable to I/Os with <i>I/O Type</i> of GPIO.

Bits	Field Name	Value	Description
10:8	I2C/MDIO Speed		The bus speed. For I²C, the I/O Widget supports the following speeds: 000b = 100 kHz 001b = 400 kHz 010b = 1 MHz All other values are reserved. For MDIO, the I/O Widget supports the following speeds: 000b = 2.441 MHz All other values are reserved. Only applicable to I/Os with <i>I/O Type</i> of I²C Bus and MDIO Bus.
7:5	I/O Type		The type of the I/O. The defined I/O types are: 000b = PMD 001b = I²C Bus 010b = MDIO Bus 011b = GPIO 100b = MDIO Address 101b = Clk In 110b = Clk Out All other values are reserved.
4:0	I/O Function / I/O Number		I/O Function: For I/Os with <i>I/O Type</i> of GPIO defines the specific use of the GPIO: 0x00 = GPIO 0x01 = RESET_N 0x02 = INT_N 0x03 = PRESENT_N 0x04 = TX_DISABLE 0x05 = MODSEL_N 0x06 = LPMODE 0x07 = TX_FAULT / TX_FAULT_0 0x08 = RX_LOS / RX_LOS_0 0x09 = RS0 / SPEED_0_1 0x0A = RS1 / SPEED_0_2 0x0B = EEPROM_WP 0x0C = LED/LED.Red 0x0D = LED.Green 0x0E = LED.Blue 0x0F = TX_DISABLE_1 0x10 = TX_FAULT_1 0x11 = RX_LOS_1 0x12 = SPEED_1_1 0x13 = SPEED_1_2 0x14 = Clk Dir 0x15 = Clk Out En 0x16 = Clk In Sel 0x17 = Safe_Boot_N All other values are reserved. I/O Number: For I/Os with <i>I/O Type</i> of PMD, I²C Bus, MDIO Bus, MDIO Address, Clk In and Clk Out, defines the index of the I/O on the node.

The I/O connections are grouped by *I/O Type*. For a given I/O connection, on the node describing the connection, the I/O Number/Function is used to identify the interface within the group. On the node identified by the *Driving Node Handle*, the driving I/O, located within the group with matching *I/O Type*, is pointed to by the *Driving I/O Number*.

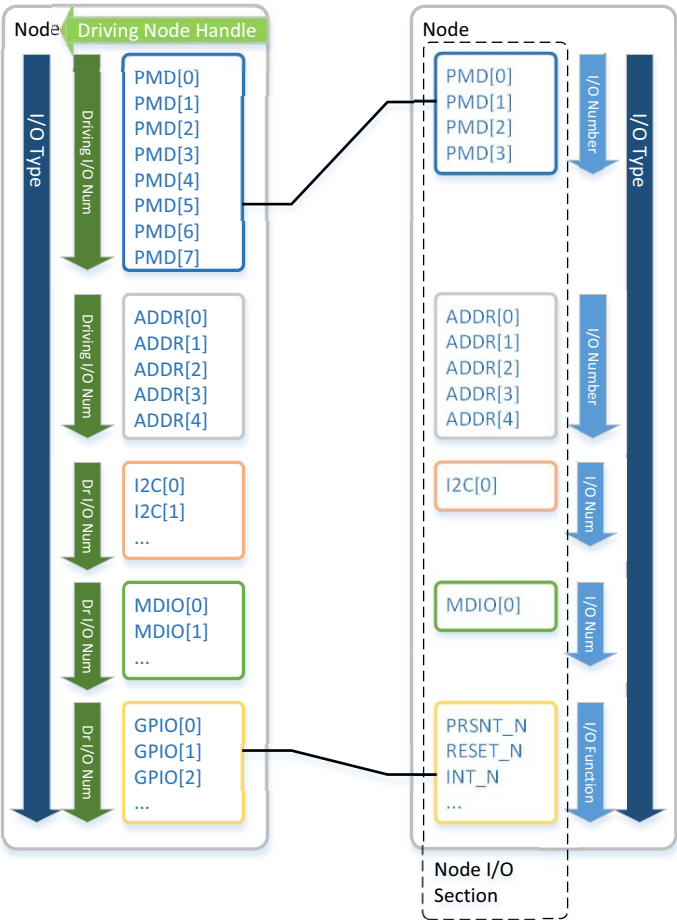


Figure 3-55. I/O Numbering Convention

3.8.6.2.5 Node Port Option Pointer Section

This section defines the structure of the section detailing the available Port Options.

Word Offset	Description	Section Reference
0x0000 + n	Port Option Pointer	3.8.6.2.5.1

The section is not accessible through the GETNODEATTR DNL action.

3.8.6.2.5.1 Node Port Option Pointer (0x0000 + n)

Bits	Field Name	Value	Description
15	Invalid		This bit marks the Port Option as valid or invalid. 0b = Port option is valid. 1b = Port option is invalid.
14:12	Reserved		Reserved.
11:0	Port Option Pointer		The Port Option's word offset relative to the beginning of the node section.

3.8.6.2.6 Node Port Option Header Section

This section defines the structure of the section detailing the available Port Options.

Word Offset	Description	Section Reference
0x0000	Adaptive NVM Global Super Configuration ID	3.8.6.2.6.1
0x0001	Reserved	3.8.6.2.6.2
0x0002	Minimum SKU	3.8.6.2.6.3
0x0003	Adaptive NVM PF-to-Port mapping Configuration ID	3.8.6.2.6.4
0x0004	PMD Count and PHY Capabilities [0] Pointer	3.8.6.2.6.5
0x0005 + n - 1	PHY Capabilities [n] Pointer	3.8.6.2.6.6

The section is not accessible through the GETNODEATTR DNL action.

3.8.6.2.6.1 Adaptive NVM Global Super Configuration ID (0x0000)

Bits	Field Name	Value	Description
15:0	Global Super Configuration ID		Adaptive NVM Global Super Configuration ID [15:0] Used to identify the MAC port configuration needed to apply to configure the controller for this Port Option (used to configure MAC port mode - 1/2/4 ports; DCB Configuration - pipe monitors, RPB thresholds; VSI/VEB allocation; etc.). Used to identify the PHY SS configuration needed to apply to configure the PHY for this Port Option (Used to configure ONPI muxes, etc.).

3.8.6.2.6.2 Reserved (0x0001)

Bits	Field Name	Value	Description
15:0	Reserved		Reserved.

3.8.6.2.6.3 Minimum SKU (0x0002)

Bits	Field Name	Value	Description
15:12	Global Super Configuration ID		Adaptive NVM Global Super Configuration ID Used to identify the adaptive NVM configuration ID for super feature ID (0xF0FD). The value here must match the number of ports defined in the port option.
11:8	Reserved		Reserved.
6:7	Required Lane Speed		Maximum lane speed required. 00b = Reserved 01b = Reserved 10b = 10G serial required. 11b = 1G serial required
5	Reserved		Reserved.
4:2	Required Bandwidth		Specifies the Required bandwidth for this port option (refers to the total bandwidth of all ports). 100b = 20 Gb/s required. 101b = 10 Gb/s required. 110b = 2 Gb/s required 111b = 1 Gb/s required others = Reserved.
1:0	Required Ports		Number of ports required: 10b = 2 ports required. 11b = 1 port required. others = Reserved.

3.8.6.2.6.4 Adaptive NVM PF-to-Port Mapping Configuration ID (0x0003)

Bits	Field Name	Value	Description
15:0	PF to Port Mapping Configuration ID		Adaptive NVM PF to Port Mapping Configuration ID This is the adaptive NVM configuration ID that selects the relevant PF-to port mapping using the adaptive NVM PF-to-port mapping feature ID.

3.8.6.2.6.5 PMD Count and PHY Capabilities Pointer (0x0004)

Bits	Field Name	Value	Description
15:12	PMD Count		The number of PMDs within the Port Option.
11:0	PHY Capabilities [0] Pointer		The PHY Capabilities Section's word offset relative to the beginning of the Node Port Option Header Section.

3.8.6.2.6.6 Capabilities Pointer (0x0005 + n-1)

Bits	Field Name	Value	Description
15:12	Reserved		Reserved.
11:0	PHY Capabilities [n] Pointer		The PHY Capabilities Section's word offset relative to the beginning of the Node Port Option Header Section.

3.8.6.2.7 Node PHY Capabilities Section

This section defines the structure of the section detailing the PHY capabilities for a PHY for a given Port Option. It is used within the PHY node, as described in

Word Offset	Description	Section Reference
0x0000	Per-Port Adaptive NVM Configuration ID	3.8.6.2.7.1
0x0001	Per-Function Adaptive NVM Configuration ID	3.8.6.2.7.2
0x0002	Per-Port/Function Adaptive NVM Feature ID	3.8.6.2.7.3
0x0003	PMD Width	3.8.6.2.7.4
0x0004	Link Options 0	3.8.6.2.7.5
0x0005	Link Options 1	3.8.6.2.7.6
0x0006	EEE Options 0	3.8.6.2.7.7
0x0007	EEE Options 1	3.8.6.2.7.8
0x0008	Host-Side PMD Capabilities 0	3.8.6.2.7.9
0x0009	Host-Side PMD Capabilities 1	3.8.6.2.7.10
0x000A	Host-Side PMD Capabilities 2	3.8.6.2.7.11
0x000B	Host-Side PMD Capabilities 3	3.8.6.2.7.12
0x000C	Line-Side PHY Capabilities (PHY Types) 0	3.8.6.2.7.13
0x000D	Line-Side PHY Capabilities (PHY Types) 1	3.8.6.2.7.14
0x000E	Line-Side PHY Capabilities (PHY Types) 2	3.8.6.2.7.15
0x000F	Line-Side PHY Capabilities (PHY Types) 3	3.8.6.2.7.16
0x0010	Line-Side PHY Capabilities (PHY Types) 4	3.8.6.2.7.17
0x0011	Line-Side PHY Capabilities (PHY Types) 5	3.8.6.2.7.18
0x0012	Line-Side PHY Capabilities (PHY Types) 6	3.8.6.2.7.19
0x0013	Line-Side PHY Capabilities (PHY Types) 7	3.8.6.2.7.20

The section is accessible through the GETNODEATTR DNL action starting at the DWord-aligned Attribute ID offset 0x4000. Attribute ID 0x4006-4009 (Line-Side PHY Capabilities 4-7) are only defined for PMDs with the *Outermost PMD* bit set.

3.8.6.2.7.1 Per-Port Adaptive NVM Configuration ID (0x0000)

Bits	Field Name	Value	Description
15:0	Per Port Configuration ID		Per-port adaptive NVM Configuration ID [15:0]. Used to identify the adaptive NVM Configuration ID needed to apply to configure the port associated with the PMD. The field applies to the Innermost PHY node only and is reserved for all other nodes.

3.8.6.2.7.2 Per-Function Adaptive NVM Configuration ID (0x0001)

Bits	Field Name	Value	Description
15:0	Per Function Configuration ID		Per-function adaptive NVM Configuration ID [15:0]. Used to identify the adaptive NVM Configuration ID needed to apply to configure the Function (PF) associated with the PMD. The field applies to the Innermost PHY node only and is reserved for all other nodes.

3.8.6.2.7.3 Per-Port/Function Adaptive NVM Feature ID (0x0002)

Bits	Field Name	Value	Description
15:8	Port Feature ID		Port Feature ID. Used to specify the adaptive NVM feature ID of the port.
7:0	Function Feature ID		Function Feature ID. Used to specify the adaptive NVM feature ID of the function.

3.8.6.2.7.4 PMD Width (0x0003)

Bits	Field Name	Value	Description
15:12	Host PMD Width		The number of lanes assigned to the host-side PMD. The host-side PMD's lanes will range from PHY Node Lane [Host PMD Lane 0] to PHY Node Lane [Host PMD Lane 0 + Host PMD Width - 1].
11:8	Host PMD Lane 0		The number of the host-side PMD's Lane 0 on the PHY Node.
7:4	Line PMD Width		The number of lanes assigned to the line-side PMD. The line-side PMD's lanes will range from PHY Node Lane [Line PMD Lane 0] to PHY Node Lane [Line PMD Lane 0 + Line PMD Width - 1].
3:0	Line PMD Lane 0		The number of the line-side PMD's Lane 0 on the PHY Node.

3.8.6.2.7.5 Link Options 0 (0x0004)

Bits	Field Name	Value	Description
15	Rx PAUSE		Controls Link Flow control Rx PAUSE default for this link. 0b = Rx PAUSE is disabled by default. 1b = Rx PAUSE is enabled by default.
14	Tx PAUSE		Controls Link Flow control Tx PAUSE default for this link. 0b = Tx PAUSE capability is disabled by default. 1b = Tx PAUSE capability is enabled by default.
13	AN Clause 37 Enable		Controls Clause 37 auto-negotiation for the line-side PMD. 0b = Clause 37 auto-negotiation is disabled. 1b = Clause 37 auto-negotiation is enabled.

Bits	Field Name	Value	Description
12	AN Clause 73 Enable		Controls Clause 73 auto-negotiation for the line-side PMD. 0b = Clause 73 auto-negotiation is disabled. 1b = Clause 73 auto-negotiation is enabled.
11	AN Clause 28 Enable		Controls Clause 28 auto-negotiation for the line-side PMD. 0b = Clause 28 auto-negotiation is disabled. 1b = Clause 28 auto-negotiation is enabled.
1	10G KR FEC Request		Controls FEC capability request for 10G KR. 0b = FEC is not requested. 1b = FEC is requested.
0	10G KR FEC Request		Controls FEC capability advertisement for 10G KR. 0b = FEC is disabled. 1b = FEC is enabled and is advertised.

3.8.6.2.7.6 Link Options 1 (0x0005)

Bits	Field Name	Value	Description
15:12	Max LPLU Speed		Max LPLU speed. Limits the maximum speed that can be enabled using Low Power Link Up (LPLU) algorithm for BASE-T PHYs. 0x2 = 1G 0x3 = 2.5G 0x4 = 5G 0x5 = 10G All other values are reserved. Note: This option is valid when LPLU is enabled (See bit 3 - LPLU enable).
11	Disable Auto Link on Startup		This bit, when set, disables the automatic link on start-up. When the port is defined as manageability port, the value of this bit is ignored and link is brought up automatically.
10	Reserved		Reserved.
9	Flow Control ASM_DIR Capability	0b	Controls Link Flow control ASM_DIR (A6) ability to advertise. 0b = 0ASM_DIR is advertised as 0. 1b = ASM_DIR is advertised as 1.
8	Flow Control PAUSE capability	0b	Controls Link Flow control PAUSE (A5) ability to advertise. 0b = PAUSE is advertised as 0. 1b = PAUSE is advertised as 1.
7	PCS Enable	1b	Controls the PCS of the PMD. - PCS is disabled, the PHY is operating in a transparent re-timer mode. - PCS is enabled.
6	LESM Auto FEC Enable	1b	Controls the FEC options for the Link Establishment State Machine (LESM). 0b = LESM uses the default FEC mode based on the current media. 1b = LESM automatically detects the link partner's FEC type. This might include FEC types not specifically supported by the spec.
5	LESM Enable	1b	Controls the Link Establishment State Machine for the PMD. 0b = LESM is disabled. 1b = LESM is enabled.
4	Low Power Mode	0b	Controls the Low Power Mode for the PHY. 0b = The device is allowed to operate at high power. 1b = The device must operate in low power mode. Pluggable modules are only allowed to operate at their lowest power level.
3	LPLU Support Enable	0b	Enables or disables the Low Power Link-up (LPLU) support for the PMD. 0b = LPLU support is disabled. 1b = LPLU support is enabled.

Bits	Field Name	Value	Description
2	Host Lane Reversal Enable	0b	Controls lane reversal on the host-side of the PMD. 0b = The lane order is normal. 1b = The lane order needs to be reversed.
1	Line Lane Reversal Enable	0b	Controls lane reversal on the line-side of the PMD. 0b = The lane order is normal. 1b = The lane order needs to be reversed.
0	Outermost PMD		Defines whether the PMD is the outermost PMD. 0b = The PMD has no outside of box connections. 1b = The PMD has outside of the box connection on the line-side.

3.8.6.2.7.7 EEE Options 0 (0x0006)

Bits	Field Name	Value	Description
15:0	EEE_TWsysTX		EEE_TWsysTX. See Section 5.4.1 .

3.8.6.2.7.8 EEE Options 1 (0x0007)

Bits	Field Name	Value	Description
15:13	Reserved		Reserved.
12	EEE Enable 5GBASE-T	0b	Controls the EEE functionality for 5GBASE-T. 0b = EEE is disabled. 1b = EEE is enabled.
11	EEE Enable 2.5GBASE-T	0b	Controls the EEE functionality for 2.5GBASE-T. 0b = EEE is disabled. 1b = EEE is enabled.
10	Reserved		Reserved
9	Reserved		Reserved
8	Reserved		Reserved
7	Reserved		Reserved
6	Reserved		Reserved
5	Reserved		Reserved
4	Reserved		Reserved
3	Reserved		Reserved
2	EEE Enable 10GBASE-T	0b	Controls the EEE functionality for 10GBASE-T. 0b = EEE is disabled. 1b = EEE is enabled.
1	EEE Enable 1000BASE-T	0b	Controls the EEE functionality for 1000BASE-T. 0b = EEE is disabled. 1b = EEE is enabled.
0	EEE Enable 100BASE-TX		Controls the EEE functionality for 100BASE-TX. 0b = EEE is disabled. 1b = EEE is enabled.

3.8.6.2.7.9 Host-Side PMD Capabilities 0 (0x0008)

Host-side PMD capabilities are mapped based on line-side link speed.

Bits	Field Name	Value	Description
15:0	64-bit Basic PHY Structure[15:0]		64-bit Basic PHY Structure[15:0]. Refer to PHY Capabilities Section.

3.8.6.2.7.10 Host-Side PMD Capabilities 1 (0x0009)

Host-side PMD capabilities are mapped based on line-side link speed.

Bits	Field Name	Value	Description
15:0	64-bit Basic PHY Structure[31:16]		64-bit Basic PHY Structure[31:16]. Refer to PHY Capabilities Section.

3.8.6.2.7.11 Host-Side PMD Capabilities 2 (0x000A)

Host-side PMD capabilities are mapped based on line-side link speed.

Bits	Field Name	Value	Description
15:0	64-bit Basic PHY Structure[47:32]		64-bit Basic PHY Structure[47:32]. Refer to PHY Capabilities Section.

3.8.6.2.7.12 Host-Side PMD Capabilities 3 (0x000B)

Host-side PMD capabilities are mapped based on line-side link speed.

Bits	Field Name	Value	Description
15:0	64-bit Basic PHY Structure[63:48]		64-bit Basic PHY Structure[63:48]. Refer to PHY Capabilities Section.

3.8.6.2.7.13 Line-Side PHY Capabilities (PHY Types) 0 (0x000C)

The following table defines the Line-Side PHY Capabilities for the outermost PMD. If the *Outermost PMD* bit is not set, the Line-Side PHY Capabilities match the Host-Side PHY Capabilities structure. See [Section 3.8.6.2.7.9](#).

Bits	Field Name	Value	Description
15:0	128-bit Extended PHY Structure[15:0]		128-bit Extended PHY Structure[15:0]. Refer to PHY Capabilities Section.

3.8.6.2.7.14 Line-Side PHY Capabilities (PHY Types) 1 (0x000D)

The following table defines the Line-Side PHY Capabilities for the outermost PMD. If the *Outermost PMD* bit is not set, the Line-Side PHY Capabilities match the Host-Side PHY Capabilities structure. See [Section 3.8.6.2.7.10](#).

Bits	Field Name	Value	Description
15:0	128-bit Extended PHY Structure[31:16]		128-bit Extended PHY Structure[31:16]. Refer to PHY Capabilities Section.

3.8.6.2.7.15 Line-Side PHY Capabilities (PHY Types) 2 (0x000E)

The following table defines the Line-Side PHY Capabilities for the outermost PMD. If the *Outermost PMD* bit is not set, the Line-Side PHY Capabilities match the Host-Side PHY Capabilities structure. See [Section 3.8.6.2.7.11](#).

Bits	Field Name	Value	Description
15:0	128-bit Extended PHY Structure[47:32]		128-bit Extended PHY Structure[47:32]. Refer to PHY Capabilities Section.

3.8.6.2.7.16 Line-Side PHY Capabilities (PHY Types) 3 (0x000F)

The following table defines the Line-Side PHY Capabilities for the outermost PMD. If the *Outermost PMD* bit is not set, the Line-Side PHY Capabilities match the Host-Side PHY Capabilities structure. See [Section 3.8.6.2.7.12](#).

Bits	Field Name	Value	Description
15:0	128-bit Extended PHY Structure[63:48]		128-bit Extended PHY Structure[63:48]. Refer to PHY Capabilities Section.

3.8.6.2.7.17 Line-Side PHY Capabilities (PHY Types) 4 (0x0010)

The following table defines the Line-Side PHY Capabilities for the outermost PMD. Not defined for PMDs with the *Outermost PMD* bit not set.

Bits	Field Name	Value	Description
15:0	128-bit Extended PHY Structure[79:64]		128-bit Extended PHY Structure[79:64]. Refer to PHY Capabilities Section.

3.8.6.2.7.18 Line-Side PHY Capabilities (PHY Types) 5 (0x0011)

The following table defines the Line-Side PHY Capabilities for the outermost PMD. Not defined for PMDs with the *Outermost PMD* bit not set.

Bits	Field Name	Value	Description
15:0	128-bit Extended PHY Structure[95:80]		128-bit Extended PHY Structure[95:80]. Refer to PHY Capabilities Section.

3.8.6.2.7.19 Line-Side PHY Capabilities (PHY Types) 6 (0x0012)

The following table defines the Line-Side PHY Capabilities for the outermost PMD. Not defined for PMDs with the *Outermost PMD* bit not set.

Bits	Field Name	Value	Description
15:0	128-bit Extended PHY Structure[111:96]		128-bit Extended PHY Structure[111:96]. Refer to PHY Capabilities Section.

3.8.6.2.7.20 Line-Side PHY Capabilities (PHY Types) 7 (0x0013)

The following table defines the Line-Side PHY Capabilities for the outermost PMD. Not defined for PMDs with the *Outermost PMD* bit not set.

Bits	Field Name	Value	Description
15:0	128-bit Extended PHY Structure[127:112]		128-bit Extended PHY Structure[127:112]. Refer to PHY Capabilities Section.

3.8.6.2.8 Node PMD Analog Section

This section describes the structure of the section detailing the analog configuration of a PHY Lane.

Word Offset	Description	Section Reference
0x0000 + n*10	1G/2.5G/5G C2C Coefficients Low	3.8.6.2.8.1
0x0001 + n*10	Polarity and 1G/2.5G/5G C2C Coefficients High	3.8.6.2.8.2
0x0002 + n*10	10G C2C Coefficients Low	3.8.6.2.8.3
0x0003 + n*10	10G C2C Coefficients High	3.8.6.2.8.4
0x0004 + n*10	10G C2M Coefficients Low	3.8.6.2.8.5
0x0005 + n*10	10G C2M Coefficients High	3.8.6.2.8.6

Note: In this context, “n” refers to the index of the node I/O connections and ranges from 0 to Line/Host Lane Count - 1.

Note: There is no standard definition for how the device must implement the coefficient values set in this section. The values set in this section of the netlist are PHY-specific, and must align with the definition conventions for the PHY that they are specifying, as well as the PHY specific DNL script that configures the device. C2C Coefficients are used for link modes directly connecting PHY devices, such as 10G-SFI-DA. C2M coefficients are used for link modes that use a module, such as 10GBASE-LR.

The section is accessible through the GETNODEATTR DNL action starting at the DWord-aligned Attribute ID offset 0x6800 for the line-side, and 0x7000 for the host-side.

3.8.6.2.8.1 1G/2.5G/5G C2C Coefficients Low (0x0000 + n*10)

Defines the transmit equalizer settings for 1G/2.5G/5G serial AUI-C2C interfaces.

Note: It is important to understand that each PHY uses its own definition for TXFFE values, including format and sign. It is required that the values saved in the netlist align with the format used by the specific PHY they are defining. Therefore, there is no standardized way that the coefficients are defined in this section. The convention used must also align with the DNL script that manages the link up process for that individual PHY. Overriding TXFFE values should only be done by an advanced user with specific knowledge of how the PHY should be configured for the given mode, such as tuning the coefficients based on conformance testing results.

Bits	Field Name	Value	Description
15:8	C[0][7:0]		Cursor value for C2C 1G/2.5G/5G speeds
7:0	C[1][7:0]		Post-Cursor value for C2C 1G/2.5G/5G speeds.

3.8.6.2.8.2 Polarity and 1G/2.5G/5G C2C Coefficients High (0x0001 + n*10)

Defines the lane polarity and transmit equalizer settings for 1G/2.5G/5G serial AUI-C2C interfaces.

Bits	Field Name	Value	Description
15	Rx Polarity	0b	Lane Rx Polarity control. 0b = Normal Rx polarity. 1b = Inverted Rx polarity.
14	Tx Polarity	0b	Lane Tx Polarity control. 0b = Normal Tx polarity. 1b = Inverted Tx polarity.
13:10	Reserved		Reserved.
9	TXFFE Override Enable		TXFFE Override Enable control. 0b = Always silicon default TXFFE values for all PHY modes. 1b = Use netlist defined TXFFE values for PHY modes based on the <i>TXFFE Override</i> bit corresponding to each mode.
8	TXFFE Override 1G//2.5/5G		TXFFE Override 1G/5G control. 0b = Use silicon default TXFFE values for PHY modes with 1G/2.5G/5G lane speeds. 1b = Use netlist defined TXFFE values for PHY modes with 1G/2.5G/5G lane speeds.
7:0	C[-1][7:0]		Pre-Cursor value for C2C 1G/2.5G/5G speeds.

3.8.6.2.8.3 10G AUI-C2C Coefficients Low (0x0002 + n*10)

Defines the transmit equalizer settings for 10G serial AUI-C2C interfaces.

Bits	Field Name	Value	Description
15:8	C[0][7:0]		Cursor value for AUI-C2C 10G speeds.
7:0	C[1][7:0]		Post-Cursor value for AUI-C2C 10G speeds.

3.8.6.2.8.4 10G AUI-C2C Coefficients High (0x0003 + n*10)

Defines the transmit equalizer settings for 10G serial AUI-C2C interfaces.

Bits	Field Name	Value	Description
15:9	Reserved		Reserved.
8	TXFEE Override 10G AUI-C2C		TXFEE Override 10G AUI-C2C control. 0b = Use silicon default TXFEE values for AUI-C2C PHY modes with 10G lane speeds. 1b = Use netlist defined TXFEE values for AUI-C2C PHY modes with 10G lane speeds.
7:0	C[-1][7:0]		Pre-Cursor value for 10G speeds.

3.8.6.2.8.5 10G AUI-C2M Coefficients Low (0x0004 + n*10)

Defines the transmit equalizer settings for 10G serial AUI-C2M interfaces.

Bits	Field Name	Value	Description
15:8	C[0][7:0]		Cursor value for AUI-C2M 10G speeds.
7:0	C[1][7:0]		Post-Cursor value for AUI-C2M 10G speeds.

3.8.6.2.8.6 10G AUI-C2M Coefficients High (0x0005 + n*10)

Defines the transmit equalizer settings for 10G serial AUI-C2M interfaces.

Bits	Field Name	Value	Description
15:9	Reserved		Reserved.
8	TXFEE Override 10G AUI-C2M		TXFEE Override 10G AUI-C2M control. 0b = Use silicon default TXFEE values for AUI-C2M PHY modes with 10G lane speeds. 1b = Use netlist defined TXFEE values for AUI-C2M PHY modes with 10G lane speeds.
7:0	C[-1][7:0]		Pre-Cursor value for C2M 10G speeds.

3.8.6.2.9 Node PMD Analog Misc Section

An extension of the Node PMD Analog Section, this section contains PHY device specific lane configuration. The section is located immediately at after the Node PMD Analog Section for both the host-side and line-side.

Word Offset	Description	Section Reference
0x0000	PMD Analog Misc Length	3.8.6.2.9.1
0x0001	Auto-Neg LESM Timeout	3.8.6.2.9.2
0x0002	1G/2.5G/5G/10G Serial AUI LESM Timeout	3.8.6.2.9.3

The section is accessible through the GETNODEATTR action starting at the DWord-aligned Attribute ID offset 0x8800 for the line-side, and 0x9000 for the host-side.

3.8.6.2.9.1 PMD Analog Misc Length (0x0000)

Bits	Field Name	Value	Description
15:8	Reserved		Reserved.
7:0	PMD Analog Misc Length		The length of the PMD analog miscellaneous section in words.

3.8.6.2.9.2 Auto-Neg LESM Timeout (0x0001)

Bits	Field Name	Value	Description
15:8	Auto-Neg LESM Timeout		Auto-negotiation LESM timeout defined in 100 ms increments.
7:0	Reserved		Reserved.

3.8.6.2.9.3 1G/2.5G/5G/10G Serial AUI LESM Timeout (0x0002)

Bits	Field Name	Value	Description
15:8	10G Serial AUI LESM Timeout		10G AUI LESM timeout defined in 100 ms increments.
7:0	1G/2.5G/5G Serial AUI LESM Timeout		1G/2.5G/5G AUI LESM timeout defined in 100 ms increments.

3.8.6.2.10 Node LED Configuration Section

This section describes the structure of the section detailing the configuration of an LED for a given Port Option. The length of the section is dependent on the number of conditions defined for the LED to indicate.

Word Offset	Description	Section Reference
0x0000	Port Option Port Affinity Low	3.8.6.2.10.1
0x0001	Port Option Port Affinity High	3.8.6.2.10.2
0x0002 + n*1	Port Option Color and Condition [n], where n = 0:3	3.8.6.2.10.3

The section is accessible through the GETNODEATTR DNL action starting at the DWord-aligned Attribute ID offset 0xA000.

3.8.6.2.10.1 Port Option Port Affinity Low (0x0000)

Defines the port affinity of the LED for a given Port Option.

If the netlist describes the LOM, the ports are referring to Physical MAC Ports. If the netlist describes a mezzanine card, the ports are referring to PHY Ports within the context of the mezzanine card. These, during the mezzanine card topology detection, are re-mapped into the Physical MAC Ports within the context of the system in which the mezzanine card was inserted. Therefore, when accessed through the GETNODEATTR DNL action, the *Port Affinity* bits always refers to Physical MAC Ports.

Bits	Field Name	Value	Description
15:8	Reserved		Reserved.
7	Port 7		If set, the LED is associated with Port 7
6	Port 6		If set, the LED is associated with Port 6
5	Port 5		If set, the LED is associated with Port 5
4	Port 4		If set, the LED is associated with Port 4
3	Port 3		If set, the LED is associated with Port 3
2	Port 2		If set, the LED is associated with Port 2
1	Port 1		If set, the LED is associated with Port 1
0	Port 0		If set, the LED is associated with Port 0

3.8.6.2.10.2 Port Option Port Affinity High (0x0001)

Bits	Field Name	Value	Description
15:14	Logic Operator	10b	The logic operator to be used between port flags. 00b = Reserved 10b = AND 10b = OR 11b = Reserved
13:9	Reserved		Reserved.
8	Port ID		If set, the LED is used for port identification based on software request.
7:0	Reserved		Reserved.

3.8.6.2.10.3 Port Option Color and Condition [n] (0x0002 + n*1)

The state of the LED is controlled by four conditions defined separately for each Port Option. The conditions are prioritized: Condition[0] being the highest in priority and Condition[3] the lowest. The true condition with the highest priority defines the Color and Blink configuration of the LED.

To allow for more flexibility, two adjacent conditions can be combined through the defined Logic Operator. If the expression Condition[i] <Operator[i]> Condition[i+1] is true, the LED is configured with the Color and Blink configuration associated with Condition[i].

Bits	Field Name	Value	Description
15:14	Reserved		Reserved.
13:11	Blink		Blink configuration for the LED. Blinking results in a cadence of 200 ms on / 200 ms off. 000b = No Blink 001b = Blink 010b = Blink on MAC Activity All other values are reserved.
10:8	Color		Defines the color of an RGB LED. For RGB LED Nodes, the allowed values are: 000b = SKIP - Skip Condition — The LED state should be driven based on the result of evaluating the next condition. 001b = RED 010b = ORANGE 011b = YELLOW 100b = GREEN 101b = BLUE 110b = PURPLE 111b = Reserved Note: Colors are only supported if the <i>Node Part Number</i> is RGB LED, the node has three I/Os defined, and they all are driven by the same PCA9585 LED controller. The selected color are mixed though driving the associated Red, Green, and Blue LEDs with the appropriate PWM signals through a PCA9585 LED controller. For monochrome LED Nodes the allowed values are as follows: 000b = SKIP - Skip Condition — The LED state should be driven based on the result of evaluating the next condition. 111 = ON All other values are reserved.
7:6	Logic Operator		The logic operator to be used between Condition[i] and Condition[i+1]. 00b = NONE 01b = AND 10b = OR 11b = Reserved The field is ignored for Condition[3].

Bits	Field Name	Value	Description
5:0	Condition		Defines the condition in which the LED should be asserted. Valid conditions are: 0x00 = FALSE 0x01 = TRUE 0x02 = LINK 0x03 = LINK_ACTIVITY 0x04 = MAC_ACTIVITY 0x05 = FILTER_ACTIVITY 0x06 = INVALID_MEDIA 0x08 = LINK_LESS_THAN_2.5G 0x09 = LINK_LESS_THAN_5G 0x0A = LINK_LESS_THAN_10G 0x12 = LINK_1G 0x13 = LINK_2.5G 0x14 = LINK_5G 0x15 = LINK_10G 0x1C = NO_LINK 0x1E:0x1F = Reserved 0x20 = TEMP_WARN_N 0x21 = TEMP_CRIT_N All other values are reserved.

3.8.6.2.11 Node Thermal Configuration Section

This section describes the structure of the section detailing the configuration of Temperature Sensor.

Word Offset	Description	Section Reference
0x0000	Min and Max Readable Value	3.8.6.2.11.1
0x0001	Tolerance and Hysteresis	3.8.6.2.11.2
0x0002	Resolution Low	3.8.6.2.11.3
0x0003	Resolution High	3.8.6.2.11.4
0x0004	Offset Low	3.8.6.2.11.5
0x0005	Offset High	3.8.6.2.11.6
0x0006	Normal Max and Warning High Thresholds	3.8.6.2.11.7
0x0007	Critical High and Fatal High Threshold	3.8.6.2.11.8
0x0008	Accuracy	3.8.6.2.11.9
0x0009	Reserved	3.8.6.2.11.10
0x000A + n*2	Temperature Sensor Port Affinity Low	3.8.6.2.11.11
0x000B + n*2	Temperature Sensor Port Affinity High	3.8.6.2.11.12

Note: In this context, “n” refers to the index of the per-port option Temperature Sensor Port Affinity and ranges from 0 to the parent node's Port Option Count - 1.

The section is accessible through the GETNODEATTR DNL action starting at the DWord-aligned Attribute ID offset 0xC000. The Temperature Sensor Port Affinity is replicated for each Port Option of the associated Parent PHY node. Only the copy with the same index as the Parent PHY's active Port Option is loaded to RAM and accessible through GETNODEATTR.

For SFP standard sensors, LM application follows the standard and should not modify parameters from the netlist. The application should not take (i.e. ignore) the critical thresholds from the netlist, where SFP standard sensor is defined. Instead, it should use thresholds as read from the module itself (SFF-8436 defines them in bytes 128-129).

3.8.6.2.11.1 Min and Max Readable Value (0x0000)

Bits	Field Name	Value	Description
15:8	Max Readable Value	0xFF	Thermal Sensor <i>maxReadable</i> value.
7:0	Min Readable Value	0x00	Thermal Sensor <i>minReadable</i> value.

3.8.6.2.11.2 Tolerance and Hysteresis (0x0001)

Bits	Field Name	Value	Description
15:8	Hysteresis	0x02	Thermal Sensor Hysteresis (°C).
7:0	Tolerance	0x01	Thermal Sensor <i>plusTolerance</i> and <i>minusTolerance</i> (°C).

3.8.6.2.11.3 Resolution Low (0x0002)

Bits	Field Name	Value	Description
15:0	Resolution [15:0]	0x0000	Lower two bytes of the Thermal Sensor resolution. The field uses a real32 representation. Note: The available resolution is 0.125 °C, but the supported value is 1 °C.

3.8.6.2.11.4 Resolution High (0x0003)

Bits	Field Name	Value	Description
15:0	Resolution [31:16]	0x3F80	Higher two bytes of the Thermal Sensor resolution. The field uses a real32 representation. Note: The available resolution is 0.125 °C, but the supported value is 1 °C.

3.8.6.2.11.5 Offset Low (0x0004)

Bits	Field Name	Value	Description
15:0	Offset [15:0]	0x0000	Lower two bytes of the Thermal Sensor offset. The field uses a real32 representation.

3.8.6.2.11.6 Offset High (0x0005)

Bits	Field Name	Value	Description
15:0	Offset [31:16]	0x0000	Higher two bytes of the Thermal Sensor offset. The field uses a real32 representation.

3.8.6.2.11.7 Normal Max and Warning High Thresholds (0x0006)

Bits	Field Name	Value	Description
15:8	Warning High		Warning High - High temperature warning threshold (°C).
7:0	Normal Max		Normal Max - Maximum normal temperature threshold (°C).

3.8.6.2.11.8 Critical High and Fatal High Threshold (0x0007)

Bits	Field Name	Value	Description
15:8	Fatal High		Fatal High - Fatal temperature alarm threshold (°C).
7:0	Critical High		Critical High - Critical temperature alarm threshold (°C).

3.8.6.2.11.9 Accuracy (0x0008)

Bits	Field Name	Value	Description
15:0	Accuracy		Thermal sensor accuracy (percentage of readable value).

3.8.6.2.11.10 Reserved (0x0009)

Bits	Field Name	Value	Description
15:0	Reserved		Reserved.

3.8.6.2.11.11 Temperature Sensor Port Affinity Low (0x000A + n*2)

This section describes the port affinity of the temperature sensor. Each bit, if set, indicates the temperature sensor is associated with the respective port.

If the netlist describes the LOM, the ports are referring to Physical MAC Ports. If the netlist describes a mezzanine card, the ports are referring to PHY Ports within the context of the mezzanine card. These, during the mezzanine card topology detection, are re-mapped into the Physical MAC Ports within the context of the system into which the mezzanine card was inserted. Therefore, when accessed through the GETNODEATTR action, the Port Affinity bits always refer to Physical MAC Ports.

Bits	Field Name	Value	Description
15:8	Reserved		Reserved.
7	Port 7		If set, the temperature sensor is associated with Port 7
6	Port 6		If set, the temperature sensor is associated with Port 6
5	Port 5		If set, the temperature sensor is associated with Port 5
4	Port 4		If set, the temperature sensor is associated with Port 4
3	Port 3		If set, the temperature sensor is associated with Port 3
2	Port 2		If set, the temperature sensor is associated with Port 2
1	Port 1		If set, the temperature sensor is associated with Port 1
0	Port 0		If set, the temperature sensor is associated with Port 0

3.8.6.2.11.12 Temperature Sensor Port Affinity High (0x000B + n*2)

Bits	Field Name	Value	Description
15:8	Sensor Location		Defines the location of the sensor. 0x00 = Not present/Not Applicable. 0x01 = Internal to ASIC. 0x02 = Near hottest part local ambient (deprecated). 0x03 = On board near main PCI connector. 0x04 = On board near rear/external connector. 0x05 = On board other. 0x06 = Off board SmartArray sensor. 0x07 = Inlet ambient on a blade mezzanine card. 0x08 = Internal to a discrete PHY. 0x10 = SFP+/SFP28 module. 0x11 = Reserved. All other values are reserved.
7:0	Reserved		Reserved.

3.8.6.2.12 Node Clock Configuration Header Section

This section defines the structure of the Node Clock Configuration Header holding the pointers to the Clock Input, Clock Output, and DPLL Configuration sections.

Word Offset	Description	Section Reference
0x0000	Node Clock Input Configuration Section Pointer	3.8.6.2.12.1
0x0001	Node Clock Output Configuration Section Pointer	3.8.6.2.12.2
0x0002	Node DPLL Configuration Section Pointer	3.8.6.2.12.3

The section is accessible through the GETNODEATTR DNL action starting at the DWord-aligned Attribute ID offset 0x0004.

3.8.6.2.12.1 Node Clock Input Configuration Section Pointer (0x0000)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15:12	Clock Input Count		The number of clock inputs on the Clock Controller node. Note: A value of 0 indicates no Node Clock Input Configuration Sections captured within the netlist.
11:0	Clock Input Configuration Section Offset		The word offset of the Node Clock Input Configuration Section relative to the beginning of the node section.

3.8.6.2.12.2 Node Clock Output Configuration Section Pointer (0x0001)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15:12	Clock Output Count		The number of clock outputs on the Clock Controller node. Note: A value of 0 indicates no Node Clock Output Configuration Sections captured within the netlist.
11:0	Clock Output Configuration Section Offset		The word offset of the Node Clock Output Configuration Section relative to the beginning of the node section.

3.8.6.2.12.3 Node DPLL Configuration Section Pointer (0x0002)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15:12	DPLL Count		The number of DPLLs within the Clock Controller node. Note: A value of 0 indicates no Node DPLL Configuration Sections captured within the netlist.
11:0	Node DPLL Configuration Section Offset		The word offset of the Node DPLL Configuration Section relative to the beginning of the node section.

3.8.6.2.13 Node Clock Input Configuration Section

Word Offset	Description	Section Reference
0x0000	Clock Input Mode	3.8.6.2.13.1
0x0001	Frequency Option Count and Clock I/O Number	3.8.6.2.13.2
0x0002	Phase Offset Compensation Lo	3.8.6.2.13.3
0x0003	Phase Offset Compensation Hi	3.8.6.2.13.4
0x0004	Max Phase Offset Compensation Lo	3.8.6.2.13.5
0x0005	Max Phase Offset Compensation Hi	3.8.6.2.13.6
0x0006 + n*2	Option[n] Frequency Lo	3.8.6.2.13.7
0x0007 + n*2	Option[n] Frequency Hi	3.8.6.2.13.8

Note: In this context, “n” refers to the index of the frequency option allowed for the clock input and ranges from 0 to Frequency Option Count - 1.

The section is accessible through the GETNODEATTR DNL action starting at the DWord-aligned Attribute ID offset 0x2800. Subsequent instances of the section are accessible at consecutive Attribute IDs: section_instance[n].attr_id_first = section_instance[n-1].attr_id_last + 1.

3.8.6.2.13.1 Clock Input Mode (0x0000)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15	eSync Ability		eSync ability of the clock input. 0b = Not eSync capable. 1b = eSync capable.
14	eSync Enable		Default eSync configuration for the clock input. 0b = Disabled 1b = Enabled
13:8	Reserved		Reserved.
7:5	Mode		Specifies the default mode of operation for the clock input. 000b = Disabled — The input is disabled during board initialization. 001b = Enabled — The input is enabled during board initialization. 011b = Auto — The input is configured based on link status of the relevant port. 111b = Silicon Default — The input is not configured until explicitly requested via an admin command. All other values are reserved.
4:0	Reserved		Reserved.

3.8.6.2.13.2 Frequency Option Count and Clock I/O Number (0x0001)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15:12	Frequency Option Count		The number of allowed frequency options for the clock input.
11:8	Default Frequency Option		The index of the default frequency option for the clock input.
7:5	Clock I/O Type		Specifies type of the clock input. 000b = Single ended GPS clock. 001b = Single ended PHY recovered clock. 010b = Single ended external clock. 011b = Single ended 1588. 100b = Differential GPS clock. 101b = Differential PHY recovered clock. 110b = Differential external clock. 111b = Differential 1588.
4:0	Clock I/O Number		The number of the clock Input I/O Connection associated with this clock input configuration. A value of 0x1F designates a clock input with no associated I/O Connections captured within the netlist.

3.8.6.2.13.3 Phase Offset Compensation Lo (0x0002)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15:0	Phase Offset Compensation Lo		Phase Offset Compensation with 1 ps resolution. Phase Offset Compensation [15:0]

3.8.6.2.13.4 Phase Offset Compensation Hi (0x0003)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15:0	Phase Offset Compensation Hi		Phase Offset Compensation with 1 ps resolution. Phase Offset Compensation [31:16]

3.8.6.2.13.5 Max Phase Offset Compensation Lo (0x0004)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15:0	Max Phase Offset Compensation Lo		Maximum Phase Offset Compensation with 1 ps resolution. Max Phase Offset Compensation [15:0]

3.8.6.2.13.6 Max Phase Offset Compensation Hi (0x0005)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15:0	Max Phase Offset Compensation Hi		Maximum Phase Offset Compensation with 1 ps resolution. Max Phase Offset Compensation [31:16]

3.8.6.2.13.7 Option[n] Frequency Lo (0x0006 + n*2)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15:0	Option[n] Frequency Lo		Option[n] Frequency with 1 Hz resolution. Option[n] Frequency [15:0]

3.8.6.2.13.8 Option[n] Frequency Hi (0x0007 + n*2)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15:0	Option[n] Frequency Hi		Option[n] Frequency with 1 Hz resolution. Option[n] Frequency [31:16]

3.8.6.2.14 Node Clock Output Configuration Section

Word Offset	Description	Section Reference
0x0000	Clock Output Mode and Source Select	3.8.6.2.14.1
0x0001	Frequency Option Count and Clock I/O Number	3.8.6.2.14.2
0x0002	Phase Offset Compensation Lo	3.8.6.2.14.3
0x0003	Phase Offset Compensation Hi	3.8.6.2.14.4
0x0004	Max Phase Offset Compensation Lo	3.8.6.2.14.5
0x0005	Max Phase Offset Compensation Hi	3.8.6.2.14.6
0x0006 + n*2	Option[n] Frequency Lo	3.8.6.2.14.7
0x0007 + n*2	Option[n] Frequency Hi	3.8.6.2.14.8

Note: In this context, “n” refers to the index of the frequency option allowed for the clock output and ranges from 0 to Frequency Option Count - 1.

The section is accessible through the GETNODEATTR DNL action starting at the DWord-aligned Attribute ID offset 0x3000. Subsequent instances of the section are accessible at consecutive Attribute IDs: section_instance[n].attr_id_first = section_instance[n-1].attr_id_last + 1.

3.8.6.2.14.1 Clock Output Mode and Source Select (0x0000)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15	eSync Ability		eSync ability of the clock output. 0b = Not eSync capable. 1b = eSync capable.
14	eSync Enable		Default eSync configuration for the clock output. 0b = Disabled 1b = Enabled
13:8	Reserved		Reserved.
7:5	Mode		Specifies the default mode of operation for the clock output. 000b = Disabled — The output is disabled during board initialization. 001b = Enabled — The output is enabled during board initialization. 011b = Force Enabled — The output is forced to enabled and any configuration attempt is denied and results in a configuration error. Note: This mode is intended for clock outputs driving the network controller's reference clock input. 111b = Silicon Default — The output is not configured until explicitly requested via an admin command. All other values are reserved.
4:0	Source Select		Default source configuration Specifies the index of the DPLL configured as a source in <i>Enabled</i> and <i>Force Enabled</i> modes.

3.8.6.2.14.2 Frequency Option Count and Clock I/O Number (0x0001)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15:12	Frequency Option Count		The number of allowed frequency options for the clock output.
11:8	Default Frequency Option		The index of the default frequency option for the clock output.
7:5	Clock I/O Type		Specifies type of the clock output. 000b = Single ended clock. 100b = Differential clock. All other values are reserved.
4:0	Clock I/O Number		The number of the clock Output I/O Connection associated with this clock output configuration. A value of 0x1F designates a clock output with no associated I/O Connections captured within the netlist.

3.8.6.2.14.3 Phase Offset Compensation Lo (0x0002)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15:0	Phase Offset Compensation Lo		Phase Offset Compensation with 1 ps resolution. Phase Offset Compensation [15:0]

3.8.6.2.14.4 Phase Offset Compensation Hi (0x0003)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15:0	Phase Offset Compensation Hi		Phase Offset Compensation with 1 ps resolution. Phase Offset Compensation [31:16]

3.8.6.2.14.5 Max Phase Offset Compensation Lo (0x0004)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15:0	Max Phase Offset Compensation Lo		Maximum Phase Offset Compensation with 1 ps resolution. Max Phase Offset Compensation [15:0]

3.8.6.2.14.6 Max Phase Offset Compensation Hi (0x0005)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15:0	Max Phase Offset Compensation Hi		Maximum Phase Offset Compensation with 1 ps resolution. Max Phase Offset Compensation [31:16]

3.8.6.2.14.7 Option[n] Frequency Lo (0x0006 + n*2)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15:0	Option[n] Frequency Lo		Option[n] Frequency with 1 Hz resolution. Option[n] Frequency [15:0]

3.8.6.2.14.8 Option[n] Frequency Hi (0x0007 + n*2)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15:0	Option[n] Frequency Hi		Option[n] Frequency with 1 Hz resolution. Option[n] Frequency [31:16]

3.8.6.2.15 Node Recovered Clock Output Configuration Section

Word Offset	Description	Section Reference
0x0000	Clock Output Mode and Source Select	3.8.6.2.15.1
0x0001	Frequency Option Count and Clock I/O Number	3.8.6.2.15.2
0x0002	Reserved	3.8.6.2.15.3
0x0003	Reserved	3.8.6.2.15.4
0x0004	Reserved	3.8.6.2.15.5
0x0005	Reserved	3.8.6.2.15.6
0x0006 + n*2	Option[n] Frequency Lo	3.8.6.2.15.7
0x0007 + n*2	Option[n] Frequency Hi	3.8.6.2.15.8

Note: In this context, “n” refers to the index of the frequency option allowed for the clock output and ranges from 0 to Frequency Option Count - 1.

The section is accessible through the GETNODEATTR DNL action starting at the DWord-aligned Attribute ID offset 0x3000. Subsequent instances of the section are accessible at consecutive Attribute IDs: section_instance[n].attr_id_first = section_instance[n-1].attr_id_last + 1.

3.8.6.2.15.1 Clock Output Mode and Source Select (0x0000)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description																		
15:8	Source Enable		Specifies the PHY Lanes allowed as recovered clock source. A lane may only be configured as a source of the clock output if the corresponding source enable bit is asserted. <table><thead><tr><th>Bit</th><th>Description</th></tr></thead><tbody><tr><td>0</td><td>Lane 0</td></tr><tr><td>1</td><td>Lane 1</td></tr><tr><td>2</td><td>Lane 2</td></tr><tr><td>3</td><td>Lane 3</td></tr><tr><td>4</td><td>Lane 4</td></tr><tr><td>5</td><td>Lane 5</td></tr><tr><td>6</td><td>Lane 6</td></tr><tr><td>7</td><td>Lane 7</td></tr></tbody></table>	Bit	Description	0	Lane 0	1	Lane 1	2	Lane 2	3	Lane 3	4	Lane 4	5	Lane 5	6	Lane 6	7	Lane 7
Bit	Description																				
0	Lane 0																				
1	Lane 1																				
2	Lane 2																				
3	Lane 3																				
4	Lane 4																				
5	Lane 5																				
6	Lane 6																				
7	Lane 7																				
7:5	Mode		Specifies the default mode of operation for the clock output. 000b = Disabled — The output is disabled during board initialization. 001b = Enabled — The output is enabled during board initialization. 010b = Force Enabled — The output is forced to enabled and any configuration attempt is denied and result in a configuration error. Note: This mode is intended for clock outputs driving the network controller's reference clock input. 011b = Auto — The output is configured based on link status of the relevant port. 111b = Silicon Default — The output is not configured until explicitly requested via an admin command. All other values are reserved.																		
4:0	Source Select		Default source configuration Specifies the PHY Lane number configure as a recovered clock source in <i>Enabled</i> and <i>Force Enabled</i> modes.																		

3.8.6.2.15.2 Frequency Option Count and Clock IO Number (0x0001)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15:12	Frequency Option Count		The number of allowed frequency options for the clock output.
11:8	Reserved		Reserved.
7:5	Clock I/O Type		Specifies type of the clock output. 000b = Single ended clock. 100b = Differential clock. All other values are reserved.
4:0	Clock I/O Number		The number of the clock Output I/O Connection associated with this clock output configuration. A value of 0x1F designates a clock output with no associated I/O Connections captured within the netlist.

3.8.6.2.15.3 Reserved (0x0002)

Bits	Field Name	Value	Description
15:0	Reserved		Reserved.

3.8.6.2.15.4 Reserved (0x0003)

Bits	Field Name	Value	Description
15:0	Reserved		Reserved.

3.8.6.2.15.5 Reserved (0x0004)

Bits	Field Name	Value	Description
15:0	Reserved		Reserved.

3.8.6.2.15.6 Reserved (0x0005)

Bits	Field Name	Value	Description
15:0	Reserved		Reserved.

3.8.6.2.15.7 Option[n] Frequency Lo (0x0006 + n*2)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15:12	Reserved		Reserved.
11:0	Option[n] Frequency Divider		The frequency divider to be configured for Option[n]. Note: The range of values and corresponding actual divide ratio is a function of the PHY Node Part Number.

3.8.6.2.15.8 Option[n] Frequency Hi (0x0007 + n*2)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15:12	Option[n] Baud Rate		The Baud Rate associated with the Option[n] 0x0 = 1.25G 0x1 = 3.125G 0x2 = 5.15625G 0x3 = 10.3125G All other values are reserved.
11:0	Reserved		Reserved.

3.8.6.2.16 Node Clock DPLL Configuration Section

Word Offset	Description	Section Reference
0x0000	DPLL Mode and Reference Select	3.8.6.2.16.1
0x0001	Source Type and EEC Mode	3.8.6.2.16.2
0x0002	Synthesizer Frequency Lo	3.8.6.2.16.3
0x0003	Synthesizer Frequency Hi	3.8.6.2.16.4
0x0004 + n	Reference Priority	3.8.6.2.16.5

Note: In this context, “n” ranges from 0 to the Clock Controller node's Clock Input Count / 2.

Note: When “n” is odd, the section must be padded with one Word of 0x0000 to ensure DWord alignment of the following sections.

The section is accessible through the GETNODEATTR DNL action starting at the DWord-aligned Attribute ID offset 0x8000. Subsequent instances of the section are accessible at consecutive Attribute IDs: section_instance[n].attr_id_first = section_instance[n-1].attr_id_last + 1.

3.8.6.2.16.1 DPLL Mode and Reference Select (0x0000)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15	Reserved		Reserved.
14	Ref Sw eSync		Reference switch on eSync failure.
13	Reserved		Reserved.
12	Ref Sw PFM		Reference switch on Precision Frequency Monitor failure.
11	Ref Sw GST		Reference switch on Guard Soak Timer failure.
10	Ref Sw CFM		Reference switch on Coarse Frequency Monitor failure.
9	Ref Sw SCM		Reference switch on Single Cycle Monitor Failure.
8	Ref Sw LOS		Reference switch on Loss of Signal.

Bits	Field Name	Value	Description
7:5	Mode		Specifies the default mode of operation for the DPLL 000b = FreeRun 001b = Force Holdover 010b = Force Lock 011b = Auto 100b = NCO 111b = Silicon Default All other values are reserved.
4:0	Reference Select		Default reference configuration Specifies the Clock Input number to be configured in Force Lock mode.

3.8.6.2.16.2 Source Type and EEC Mode (0x0001)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15:8	Reserved		Reserved.
7:5	Source Type		Specifies type of associated reference clock inputs. 000b = Single ended GPS clock. 001b = Single ended PHY recovered clock. 010b = Single ended external clock. 011b = Single ended 1588. 100b = Differential GPS clock. 101b = Differential PHY recovered clock. 110b = Differential external clock. 111b = Differential 1588.
4	Reserved		Reserved.
3:0	EEC Mode		Ethernet Equipment Clock Mode. Specifies the default clock quality requirement. 0xA = QL-EEC2/QL-SEC 0xB = QL-EEC1/QL-ST3 0xF = QL-DUS (i.e. Unknown) All other values are reserved.

3.8.6.2.16.3 Synthesizer Frequency Lo (0x0002)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15:0	Synthesizer Frequency Lo		Synthesizer Frequency with 1 Hz resolution. Synthesizer Frequency [15:0]

3.8.6.2.16.4 Synthesizer Frequency Hi (0x0003)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15:0	Synthesizer Frequency Hi		Synthesizer Frequency with 1 Hz resolution. Synthesizer Frequency [31:16]

3.8.6.2.16.5 Reference Priority (0x0004 + n)

Applicable only to nodes with Node Type of Clock Controller.

Bits	Field Name	Value	Description
15:13	Reserved		Reserved.
12:8	Reference Priority[n*2+1]		Default reference priority for Clock Input [n*2+1]. A value of 0 designates the highest priority and higher values correspond to lower priorities. Note: A value of 0x1F invalidates the corresponding reference clock input for the DPLL.
7:5	Reserved		Reserved.
4:0	Reference Priority[n*2+0]		Default reference priority for Clock Input [n*2+0].

3.8.6.2.17 Node Parent Section

This section describes the structure of a node's parent node section. LED and Temperature Sensor nodes might be associated with a PHY node as they might contain different configuration for all Port Options of the specified PHY node. Used by firmware to select which configuration section to load.

Word Offset	Description	Section Reference
0x0000	Parent Node Handle	3.8.6.2.17.1
0x0001	Reserved	3.8.6.2.17.2

3.8.6.2.17.1 Parent Node Handle (0x0000)

Bits	Field Name	Value	Description
15:11	Reserved		Reserved.
10	Parent Valid	1b	Indicates whether the Parent Node ID is valid. 0b = The node does not have a parent node. The <i>Parent Node Handle</i> field should be ignored. 1b = The node is associated with a parent node. The parent node is identified by the <i>Parent Node Handle</i> .
9:0	Parent Node Handle		The node handle of the parent node.

3.8.6.2.17.2 Reserved (0x0001)

Bits	Field Name	Value	Description
15:0	Reserved		Reserved.

3.8.6.2.18 Node Scratch Section

This chapter describes the structure of the node scratch section. Used by DNL scripts to store node related temporary variables, this section is only available at run-time and is not stored in the netlist binary. The exact use of the scratch section is subject to DNL script implementation and it is beyond the scope of this spec. The section shall be initialized to 0x0000 on POR, EMPR, and DNL Reset.

Word Offset	Description	Section Reference
0x0000	Scratch 0	3.8.6.2.18.1
0x0001	Scratch 1	3.8.6.2.18.2
0x0002	Scratch 2	3.8.6.2.18.3
0x0003	Scratch 3	3.8.6.2.18.4
0x0004	Scratch 4	3.8.6.2.18.5
0x0005	Scratch 5	3.8.6.2.18.6

The section is accessible through the GETNODEATTR and SETNODEATTR DNL action starting at the DWord-aligned Attribute ID offset 0x3800.

3.8.6.2.18.1 Scratch 0 (0x0000)

Bits	Field Name	Value	Description
15:0	Reserved		Reserved.

3.8.6.2.18.2 Scratch 1 (0x0001)

Bits	Field Name	Value	Description
15:0	Reserved		Reserved.

3.8.6.2.18.3 Scratch 2 (0x0002)

Bits	Field Name	Value	Description
15:0	Reserved		Reserved.

3.8.6.2.18.4 Scratch 3 (0x0003)

Bits	Field Name	Value	Description
15:0	Reserved		Reserved.

3.8.6.2.18.5 Scratch 4 (0x0004)

Bits	Field Name	Value	Description
15:0	Reserved		Reserved.

3.8.6.2.18.6 Scratch 5 (0x0005)

Bits	Field Name	Value	Description
15:0	Reserved		Reserved.

3.8.6.2.19 PHY Node

This section describes the block structure of a PHY node. The block contains several sections of variable length: the Node I/O Section, host-side and line-side PMD Analog Sections, host-side and line-side PMD Analog Misc Sections, Port Option Pointer Section, Port Option Header Sections, and the Node PHY Capabilities Sections.

A PHY can support multiple Port Options. The number of the supported Port Options is defined by the *Port Option Count* field in the Node Header Section. The netlist captures the Port Options in order of decreasing priority.

For each of these options the Node PHY Options Header section captures the number of PMDs available. For each of the PMDs in a given Port Option the node includes a Node PHY Capabilities Section.

The Global Configuration ID of the active Port Option on the innermost PHY node identifies the required port configuration for the current Port Option. The active Port Option of all connected PHY nodes is selected as a function of the active Port Option of the innermost PHY. The active Port Option of the PHY node identified by the Parent PHY Handle of an LED node determines the active Port Option of the LED node.

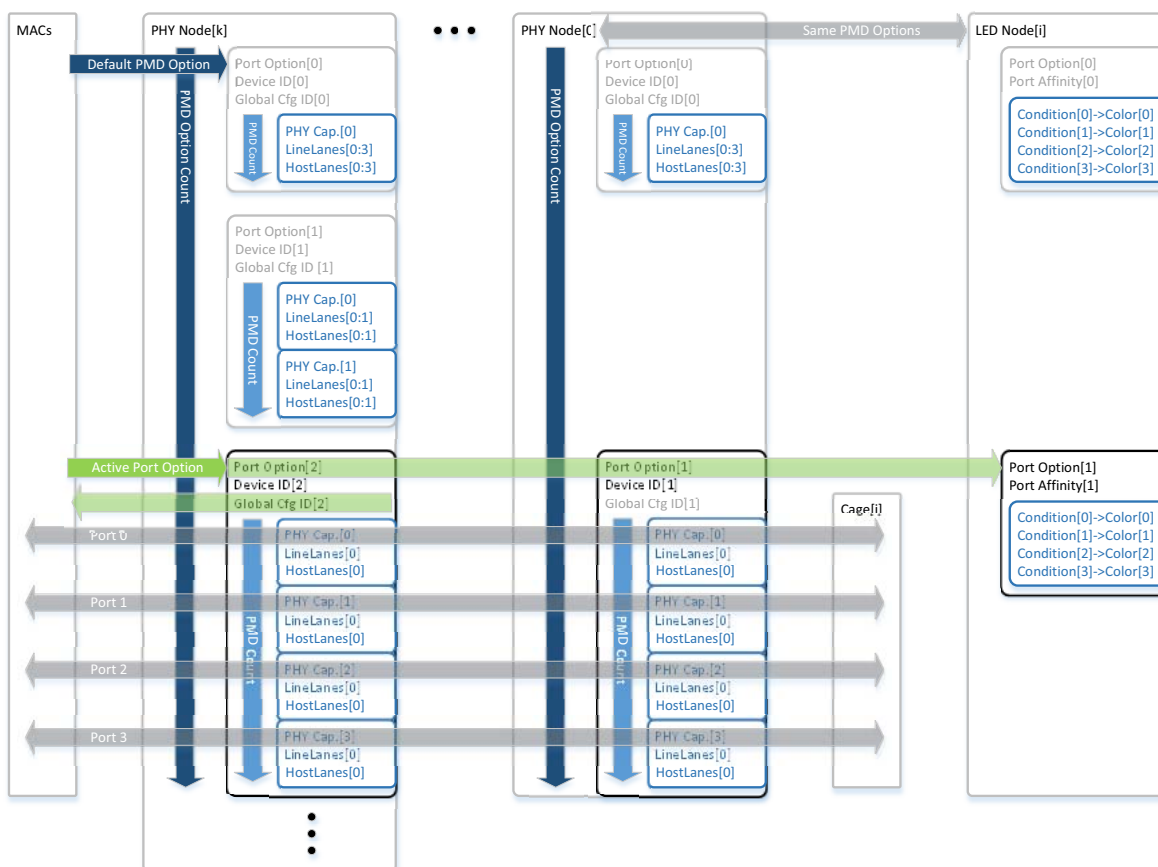


Figure 3-56. Port Options

Word Offset	Attribute ID	Description	Section Reference
0x0000	0x0000	Node Header Section	3.8.6.2.3
0x0008	0x2000	Node I/O Section	3.8.6.2.4
	0x6800	PMD Line Analog Section	3.8.6.2.8
	0x8800	PMD Line Analog Misc Section	3.8.6.2.9
	0x7000	PMD Host Analog Section	3.8.6.2.8
	0x9000	PMD Host Analog Misc Section	3.8.6.2.9
	N/A	Port Option Pointer Section	3.8.6.2.5
	N/A	Port Option Header Section [0:Port Option Count]	3.8.6.2.6
	0x4000	PHY Capabilities Section [0:PMD Count - 1][0:Port Option Count]	3.8.6.2.7

The Node I/O Section of external PHYs should define the following I/Os: I²C or MDIO, Int_N, Reset_N, PMD[0:n]. Additionally if the *Bus Address Type* defined in the Node Header Section is Relative Address ADDR[0:4] can also be defined.

3.8.6.2.20 GPIO Controller Node

This section describes the block structure of a GPIO Controller node. The block contains one section of variable length: the Node I/O Section.

Word Offset	Attribute ID	Description	Section Reference
0x0000	0x0000	Node Header Section	3.8.6.2.3
0x0008	0x2000	Node I/O Section	3.8.6.2.4
N/A	0x3800	Node Scratch Section	3.8.6.2.18

The Node I/O Section should define the following I/Os: I²C, Int_N, and Reset_N.

3.8.6.2.21 MUX Controller Node

This section describes the block structure of an MUX Controller node. The block contains one section of variable length: the Node I/O Section.

Word Offset	Attribute ID	Description	Section Reference
0x0000	0x0000	Node Header Section	3.8.6.2.3
0x0008	0x2000	Node I/O Section	3.8.6.2.4
N/A	0x3800	Node Scratch Section	3.8.6.2.18

The Node I/O Section should define the following I/Os: I²C and Reset_N.

3.8.6.2.22 LED Controller Node

This section describes the block structure of an LED Controller node. The block contains one section of variable length: the Node I/O Section.

Word Offset	Attribute ID	Description	Section Reference
0x0000	0x0000	Node Header Section	3.8.6.2.3
0x0008	0x2000	Node I/O Section	3.8.6.2.4
N/A	0x3800	Node Scratch Section	3.8.6.2.18

The Node I/O Section should define the following I/Os: I²C and Reset_N.

3.8.6.2.23 LED Node

This section describes the block structure of an LED node. The block contains two or more sections of variable length: the Node I/O Section and the Node LED Configuration Section.

An LED node can have one or three I/O connections. The number is reflected by the *I/O Count* field of the Node Header Section (Section 3.8.6.2.3). If the LED is a monochrome indicator, the I/O Section defines a single I/O (*I/O Type* LED) and if the LED is an RGB color indicator then the I/O Section defines three I/Os (*I/O types* LED.Red, LED.Green, and LED.Blue).

The LED can have a different configuration for each of the Port Options of the PHY it is associated with. Therefore the LED Node block contains a Node LED Configuration Section for each of the Port Options of the PHY with which the LED is associated. When the Parent Node Handle is valid, the number of the Port Options captured in the Node Header Section (Section 3.8.6.2.3) must match the number of Port Options defined for the PHY node with which the LED is associated. Otherwise, the number of Port Options in the node header should be 1.

Word Offset	Attribute ID	Description	Section Reference
0x0000	0x0000	Node Header Section	3.8.6.2.3
0x0008	0x2000	Node I/O Section	3.8.6.2.4
0x000A / 0x000E	N/A	Node Parent Section	3.8.6.2.17
0x000C + n*6 / 0x00010 + n*6	0xA000	Node LED Configuration Section	3.8.6.2.10
N/A	0x3800	Node Scratch Section	3.8.6.2.18

Note: In this context, “n” refers to the index of the per port option Temperature Sensor Port Affinity and ranges from 0 to the parent node's Port Option Count - 1.

3.8.6.2.24 Temperature Sensor Node

This section describes the block structure of a Temperature Sensor node. The block contains one section of variable length: the Node I/O Section.

Word Offset	Attribute ID	Description	Section Reference
0x0000	0x0000	Node Header Section	3.8.6.2.3
0x0008	0x2000	Node I/O Section	3.8.6.2.4
0x000C	N/A	Node Parent Section	3.8.6.2.17
0x0010	0xC000	Node Thermal Configuration Section	3.8.6.2.11

The Temperature Sensor node can have a different configuration for each of the Port Options of the PHY with which it is associated. Therefore, the Temperature Sensor node block contains a Node Thermal Configuration Section for each of the Port Options of the PHY with which the Temperature Sensor is associated. When the parent node handle is valid, the number of the Port Options captured in the Node Header Section (Section 3.8.6.2.3) must match the number of Port Options defined for the PHY node with which the Temperature Sensor is associated. Otherwise, the number of Port Options in the node header should be 1.

The Node I/O Section should define the following I/Os: I²C, Int_N, and Reset_N.

3.8.6.2.25 ID EEPROM Node

This section describes the block structure of an ID EEPROM node. The block contains one section of variable length: the Node I/O Section.

Word Offset	Attribute ID	Description	Section Reference
0x0000	0x0000	Node Header Section	3.8.6.2.3
0x0008	0x2000	Node I/O Section	3.8.6.2.4

The Node I/O Section should define the following I/Os: I²C and EEPROM_WP.

3.8.6.2.26 Cage Node

This section describes the block structure of a Cage node. The block contains one section of variable length: the Node I/O Section.

Word Offset	Attribute ID	Description	Section Reference
0x0000	0x0000	Node Header Section	3.8.6.2.3
0x0008	0x2000	Node I/O Section	3.8.6.2.4
N/A	0x3800	Node Scratch Section	3.8.6.2.18

The Node Header Section should indicate a *Bus Type* of I²C and the *Bus Address* should be 0xA0.

For an SFP+ or SFP28 cage the following I/Os must be defined: I²C, Present_N. Optionally the following I/Os could be also defined: TX_Disable, RS_0, RS_1.

For an SFP-DD cage the following I/Os must be defined: I²C, Present_N. Optionally the following I/Os could be also defined: TX_Disable_0, TX_Disable_1, SPEED_0_1, SPEED_0_2, SPEED_1_1, SPEED_1_2.

3.8.6.2.27 Clock Controller Node

This section describes the block structure of a Clock Controller node. The block contains several sections of variable length: the Node I/O Section, the Node Clock Input Configuration Sections, the Node Recovered Clock/Clock Output Configuration Sections, and Node Clock DPLL Configuration Sections.

Word Offset	Attribute ID	Description	Section Reference
0x0000	0x0000	Node Header Section	3.8.6.2.3
0x0008	0x2000	Node I/O Section	3.8.6.2.4
	N/A	Node Parent Section	3.8.6.2.17
	0x0004	Node Clock Configuration Header Section	3.8.6.2.12
	0x2800	Node Clock Input Configuration Section Note: Only present on clock controller nodes with a part number of a DPLL (i.e. ZL30632/ZL80032, etc.). On a clock controller with a node part number of a PHY the corresponding pointer in the Clock Configuration Header Section needs to have a DPLL count of 0	3.8.6.2.13
	0x3000	Node Recovered Clock/Clock Output Configuration Sections Part Num Section 0x24 ZL30632/ZL80032 — Node Clock Output Configuration Sections. 0x31 C827 — Node Recovered Clock Output Configuration Sections. 0x49 E610 PHY — Node Recovered Clock Output Configuration Sections.	3.8.6.2.15 3.8.6.2.14
	0x8000	Node Clock DPLL Configuration Sections Note: Only present on clock controller nodes with a part number of a DPLL (i.e. ZL30632/ZL80032, etc.). On a clock controller with a node part number of a PHY the corresponding pointer in the Clock Configuration Header Section needs to have a DPLL count of 0	3.8.6.2.16
N/A	0x3800	Node Scratch Section	3.8.6.2.18

3.8.6.2.28 Clock MUX Node

This section describes the block structure of a Clock MUX node. The block contains one section of variable length: the Node I/O Section.

Word Offset	Attribute ID	Description	Section Reference
0x0000	0x0000	Node Header Section	3.8.6.2.3
0x0008	0x2000	Node I/O Section	3.8.6.2.4

3.8.6.2.29 GPS Node

This section describes the block structure of a GPS node. The block contains one section of variable length: the Node I/O Section.

Word Offset	Attribute ID	Description	Section Reference
0x0000	0x0000	Node Header Section	3.8.6.2.3
0x0008	0x2000	Node I/O Section	3.8.6.2.4

3.8.6.3 Constraints and Conventions

The topology netlist describes design specific configuration and connections between various motherboard or mezzanine card components, referred to as nodes.

The Topology Engine imposes the following constraints:

3.8.6.3.1 Netlist Size Constraints

- The maximum size of the netlist supported in RAM is 12 KB including the motherboard and all potential mezzanine cards. For PHY and LED nodes this includes only the currently active Port Option.
- The maximum size of the motherboard netlist in the Flash is 28 KB.
- The maximum size of the mezzanine card netlist in the IDEEPROM is 4 KB.
- The maximum size of the mezzanine card netlist in RAM is 2 KB. For all PHY and LED nodes this includes only the currently active Port Option.

3.8.6.3.2 Node Constraints

- The maximum number of nodes supported on a motherboard is 128.
Note: The netlist infrastructure could allow for up to 512 nodes on a motherboard.
- The maximum depth of cascaded PHY nodes (PHY chain length) is 4.
- The maximum number of nodes of Mezzanine Connector type supported on a motherboard is 5.
Note: The netlist infrastructure could allow for up to 8 mezzanine cards.
- The maximum number of nodes of Mezzanine Connector type supported on a mezzanine card is 1.
- The maximum number of nodes supported on a mezzanine card is 32.
Note: The netlist infrastructure could allow for up to 64 nodes on a mezzanine card.

3.8.6.3.3 Node I/O Constraints

- The maximum number of I/O connections supported on the motherboard is 384.
- The maximum number of I/O connections supported on a node is 16.
- The maximum number of I/O connections of a given I/O type to a node is 64.
- The maximum number of I/O connections with I/O Type of PMD supported on a motherboard is 40.
- The maximum number of I/O connections with I/O Type of PRESENT_N supported on a motherboard is 20.
- The maximum number of I/O connections with I/O Type of INT_N supported on a motherboard is 16.
- Each node can have up to one I/O connection with I/O type GPIO and a given I/O Function / I/O Number.
- All I/O connections to a specific driving I/O on any given node must have matching configuration. For example, all I/O connections with *I/O Type* of GPIO and *I/O Function* of RESET_N connecting to the same driving GPIO input must have matching Polarity and Default Value configuration.

Note: Ethernet Topology Tool (ETT) enforces that all I/O connections to a specific driving I/O on any given node must have matching configuration.

- With the exception of I/O connections with *I/O Type* of I²C Bus, MDIO Bus, and GPIO with *I/O Function* of RESET_N, RS0 / SPEED_0_1, RS1 / SPEED_0_2, SPEED_1_1, or SPEED_1_2 must be dedicated point to point connections

Note: Ethernet Topology Tool (ETT) enforces that all I/O Connections with the exception of reset signals of I²C attached nodes are dedicated point to point connections.

- If a node is accessible through an I²C or MDIO, the connection must be described by the first entry in the I/O connections.
- On each board, the I²C attached MUX Controllers, GPIO Controllers, and LED Drivers must have a shared reset signal. Each of these nodes must have an I/O connection with *I/O Type* of GPIO and *I/O Function* of RESET_N pointing to the same driving I/O on the same driving node.
- PHY nodes must not share their reset signal with other devices. The driving I/O on the driving node pointed to by a PHY node's I/O connection with *I/O Type* of GPIO and *I/O Function* of RESET_N must not have more than one connection to it.
- The maximum number of MDIO/I²C I/Os on the E610 GPIO node is 16.
- The maximum number of MDIO/I²C I/Os on a Mux node is 62.
- The maximum number of I/O connections supported on a mezzanine card is 64.
- The maximum number of I/O connections with *I/O Type* of PMD supported on a mezzanine card is 8.
- The maximum number of I/O connections with *I/O Type* of PRESENT_N supported on a mezzanine card is 4.
- The maximum number of I/O connections with *I/O Type* of INT_N supported on a mezzanine card is 4.
- Mezzanine cards that implement more than one PHY, should not share their reset signal.

3.8.6.3.4 Node I/O Conventions

3.8.6.3.4.1 E610 PHY

- Pins RX_L0_p, RX_L0_n, TX_L0_p and TX_L0_n are referred to as PMD[0].
- Pins RX_L1_p, RX_L1_n, TX_L1_p and TX_L1_n are referred to as PMD[1].

3.8.6.3.4.2 E610 GPIO

- Pins MDC[0:4]/SCL[0:4], MDIO[0:4]/SDA[0:4] are referred to as MDIO[0:4] or I²C[0:4] depending on the I/O Type of the connected I/O connection.
- Pins SDP[0:31] are referred to as GPIO[0:31].

3.8.6.3.4.3 PCA9545A

- Pins SD0 and SC0 defined in Chapter 6 of the PCA9545A Datasheet are referred to as I²C[0].
- Pins SD1 and SC1 defined in Chapter 6 of the PCA9545A Datasheet are referred to as I²C[1].
- Pins SD2 and SC2 defined in Chapter 6 of the PCA9545A Datasheet are referred to as I²C[2].

- Pins SD3 and SC3 defined in Chapter 6 of the PCA9545A Datasheet are referred to as I²C[3].

3.8.6.3.4.4 PCA9575

- Pins P0_0 through P0_7 defined in Table 3 of the PCA9575 data-sheet are referred to as GPIO[0:7].
- Pins P1_0 through P1_7 defined in Table 3 of the PCA9575 data-sheet are referred to as GPIO[8:15].

3.8.6.3.4.5 PCA9685

- Pins LED0 through LED15 defined in Table 3 of the PCA9685 Datasheet are referred to as GPIO[0:15].

3.8.6.3.4.6 ZL30632/ZL80032 Clock Controller

- Pin REF0P defined in the DPLL Datasheet is referred to as Clk In [0].
- Pin REF0N defined in the DPLL Datasheet is referred to as Clk In [1].
- Pin REF1P defined in the DPLL Datasheet is referred to as Clk In [2].
- Pin REF1N defined in the DPLL Datasheet is referred to as Clk In [3].
- Pin REF2P defined in the DPLL Datasheet is referred to as Clk In [4].
- Pin REF2N defined in the DPLL Datasheet is referred to as Clk In [5].
- Pin REF3P defined in the DPLL Datasheet is referred to as Clk In [6].
- Pin REF3N defined in the DPLL Datasheet is referred to as Clk In [7].
- Pin REF4P defined in the DPLL Datasheet is referred to as Clk In [8].
- Pin REF4N defined in the DPLL Datasheet is referred to as Clk In [9].
- Pin OUT0N defined in the DPLL Datasheet is referred to as Clk Out [0].
- Pin OUT0P defined in the DPLL Datasheet is referred to as Clk Out [1].
- Pin OUT1P defined in the DPLL Datasheet is referred to as Clk Out [2].
- Pin OUT1N defined in the DPLL Datasheet is referred to as Clk Out [3].
- Pin OUT2P defined in the DPLL Datasheet is referred to as Clk Out [4].
- Pin OUT2N defined in the DPLL Datasheet is referred to as Clk Out [5].
- Pin OUT3P defined in the DPLL Datasheet is referred to as Clk Out [6].
- Pin OUT3N defined in the DPLL Datasheet is referred to as Clk Out [7].
- Pin OUT4P defined in the DPLL Datasheet is referred to as Clk Out [8].
- Pin OUT4N defined in the DPLL Datasheet is referred to as Clk Out [9].
- Pin OUT5P defined in the DPLL Datasheet is referred to as Clk Out [10].
- Pin OUT5N defined in the DPLL Datasheet is referred to as Clk Out [11].
- Pin OUT6P defined in the DPLL Datasheet is referred to as Clk Out [12].
- Pin OUT6N defined in the DPLL Datasheet is referred to as Clk Out [13].
- Pin OUT7P defined in the DPLL Datasheet is referred to as Clk Out [14].

- Pin OUT7N defined in the DPLL Datasheet is referred to as Clk Out [15].
- Pin OUT8P defined in the DPLL Datasheet is referred to as Clk Out [16].
- Pin OUT8N defined in the DPLL Datasheet is referred to as Clk Out [17].
- Pin OUT9P defined in the DPLL Datasheet is referred to as Clk Out [18].
- Pin OUT9N defined in the DPLL Datasheet is referred to as Clk Out [19].

3.8.6.3.4.7 E610 Clock Controller

- Pin E610 SYNCE CLK 0 is referred to as Clk Out [0].
- Pin E610 SYNCE CLK 1 is referred to as Clk Out [1].

3.8.6.3.4.8 Generic GPS

- Pin I2C is referred to as I2C.
- Pin RESET_N is referred to as RESET_N.
- Pin SAFE_BOOT_N is referred to as SAFE_BBOT_N.
- Pin INT_N is referred to as INT_N.
- Pin PRSNT_N is referred to as PRSNT_N.
- Pin Clk Out 0 is referred to as Clk Out [0].
- Pin Clk Out 1 is referred to as Clk Out [1].
- Pin E610 SYNCE CLK 0 is referred to as Clk Out [0].
- Pin E610 SYNCE CLK 1 is referred to as Clk Out [1].

3.8.6.3.4.9 Generic Clock MUX

The Generic Clock MUX node is dedicated for describing muxing options available on external clock input and output connections.

The node only captures the internal facing pins and connections and the external connections are omitted.

- Clk Dir is referred to as Clk Dir.
- Clk Out En is referred to as Clk Out En.
- Clk In Sel is referred to as Clk In Sel.
- Clk Out 0 is referred to as Clk Out [0.]
- Clk In 0 is referred to as Clk In [0].

Figure 3-57 shows the assumed internal topology of the clock mux node.

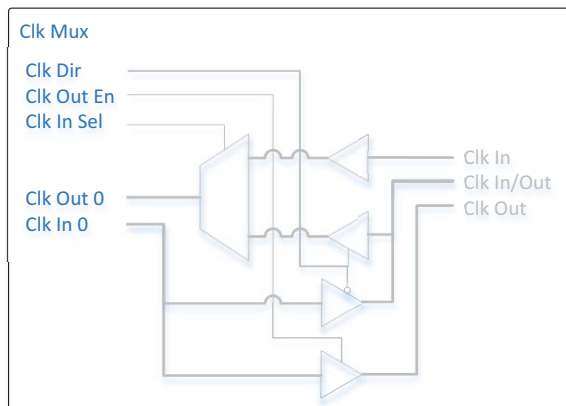


Figure 3-57. Clock MUX I/O Conventions

Note: The Generic Clock MUX nodes are not configured by the Link Manager and as a result the node may also be used to abstract other clock muxing topologies.

3.8.6.3.5 PHY Node Constraints

- The maximum number of lanes on the line or host side of a PHY node is 2.
- The maximum number of Port Options supported for a PHY node is 16.
- The E610 defines one PHY node for the two lanes of the PHY Subsystem.
- The E610 allows for one E610 PHY node in each netlist. This allows for a total of 2 line side lanes, therefore the Host Lane Count is 0, the Node PMD Host Analog Section Pointer is ignored and the corresponding Node PMD Analog Section does not exist.
- The netlist must capture the PHY nodes in a well-defined order. All PHY nodes must be preceded by the PHY nodes that it has host side connections to (I/O Connections with I/O Type of PMD). This results in the innermost PHY nodes needing to be captured first.
- All I/O connections with I/O Type of PMD from any PHY node must connect to the same node. See the examples in [Figure 3-58](#) and [Figure 3-59](#):

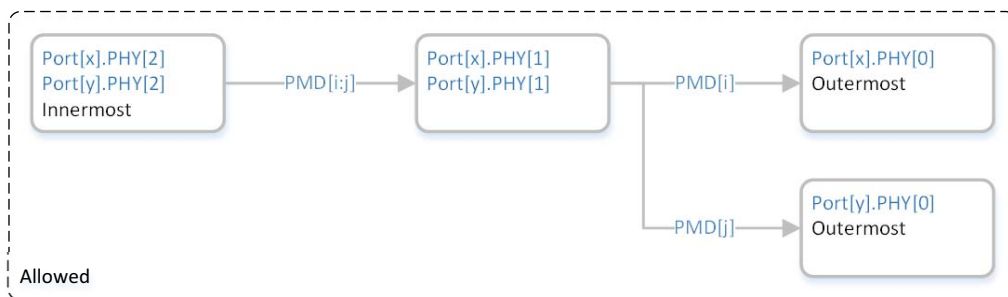


Figure 3-58. Allowed PMD Connections

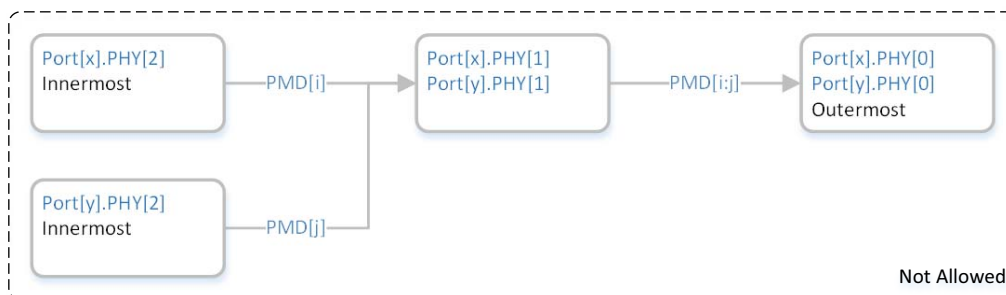


Figure 3-59. Not Allowed PMD Connections

3.8.6.3.6 GPIO Controller Node Constraints

- The E610 allows one E610 CFIO node in each netlist.
- The netlist must capture the GPIO nodes in a well-defined order. The first GPIO controller captured by the netlist must be the E610 GPIO.

3.8.6.3.7 LED Node Constraints

- All I/O connections of an LED node must be to the same driving node.
- The parent PHY node of a LED node, should be described in the netlist prior to the description of the LED node.

3.8.6.3.8 Temperature Node Constraints

- The parent PHY node of a Temperature node, should be described in the netlist prior to the description of the Temperature node.

3.8.6.3.9 Cage Node Constraints

- The I/O connection with *I/O Type* of I²C must point to an I²C interface dedicated to the cage. All cages must have dedicated I²C interfaces.

3.8.7 Link Topology Access Allowlist

3.8.7.1 Overview

To control which sections of the CSR register space are writable by the DNL script engine or the software driver the below defined NVM Allowlist section defines a set of CSR blocks that may be modified. Those CSRs can be directly mapped into the E610 CSR space, mapped into the IOSF-SB space that is accessible by the E610, or mapped into I²C or MDIO devices that are accessed by E610 firmware. Single level of indirection is also controlled by the Allowlist.

The same Allowlist is also used to monitor access by host admin commands. The monitoring is done for direct CSR access, as well as registers that are accessed using the SB-IOSF fabric via transactions initiated through the Neighbor Device Request admin command. In addition registers that are mapped on external I²C or MDIO buses are also being monitored by the same Allowlist.

The following access rules are used:

- Check that the sideband target is added to the "Allowed Sideband Target" list. When the access is performed via the "SB-IOSF" fabric, the target must be allowed. There are up to 16 SB-IOSF targets, listed in Table 3-33 and the NVM word: "Allowed SB target" within EMP SR setting module contain 1 bit for each target. This check is done regardless of other parameters of the transaction, e.g. for all read and write transactions with any other parameters. Firmware should return EINVAL error code when the sideband target is not in the allowed sideband target list
- Check the size of the sideband transaction. When the "Block 64 bits transactions" NVM bit is set, all accesses that are read or write of 64 bits are blocked. A 64 bit transaction is identified with SBE != 0. Therefore, when the host generates a transaction with SBE != 0 and the NVM bit is set, the firmware should respond with EINVAL error code
- Check that the opcode is enabled at the "Allowed opcodes" NVM word. There are up to 16 SB-IOSF opcodes, listed in Table 3-34 and the NVM word: "Allowed opcodes" within EMP SR setting module contain 1 bit for each target. This check is done regardless of other parameters of the transaction, e.g. for all read and write transactions with any other parameters. Firmware should return EINVAL error code when the sideband target is not in the allowed sideband target list

The NVM Allowlist section includes four sub-sections with Allowlists, E610 CSRs Allowlist, IOSF-SB CSRs of sideband devices Allowlist, I²C devices Allowlist, and MDIO devices Allowlist.

Note: The access control can be disabled by setting the number of defined regions sections to 0. In this case all write access is allowed. NVM image with zero entries in the Allowlist is only allowed for debug images. All production NVMs must have at least one entry in the Allowlist.

Note: If the CSR Allowlist section is not present in the NVM at all, the firmware will allow read only access to the CSR registers.

Table 3-233 describes the Allowlist NVM section:

Table 3-233. Allowlist NVM Section Structure

Subsection	Field	Address [Words]	Description
	Section Length	0x0	The total length of the NVM section.
MAC CSR Allowlist	Number of Regions	0x1	Number of accessible CSR blocks in E610 CSRs allowlist (B1).
	Region[0]	0x2:0x11	See Table TBD.
	...		
	Region[7]	0x72:0x81	
IOSF-SB Allowlist	Number of Regions	0x82	Number of accessible CSR blocks in IOSF-SB CSRs allowlist (B2).
	Reserved	0x83	
	Region[0]	0x84:0x93	See Table TBD.
	...		
	Region[31]	0x274:0x283	
I ² C Devices Allowlist	Number of Regions	0x284	Number of Accessible CSR blocks in I ² C devices allowlist (B3).
	Reserved	0x285	
	Region[0]	0x286:0x295	See Table TBD.
	...		
	Region[31]	0x476:0x485	
MDIO Devices Allowlist	Number of Regions	0x486	Number of Accessible CSR blocks in MDIO devices allowlist (B4).
	Reserved	0x487	
	Region[0]	0x488:0x497	See Table TBD.
	...		
	Region[31]	0x678:0x687	

Each allowlist item either describes a single address range (region), or describes an indirect region.

Each item occupies 16 words of 16 bits each to allow 32 bits mask and compare values. The allowlist item structure format is detailed in [Table 3-234](#):

Table 3-234. Allowlist NVM Subsection Structure

Field Name	Address [Words]	Description
Target ID	0	See Section 3.8.7.4 .
Region Control	1	See Section 3.8.7.5 .
Start Address	2:4	When Region Type is 00: Word Description 2:4 48 bits start address for the region. When Region Type is 01 or 10: Word Description 2:4 48 bits address of the control gateway register.
Reserved	5	Reserved.
End Address/Parameters	6:9	When Region Type is 00: Word Description 6:8 48 bits end address for the region 9 Reserved When Region Type is 01: Word Description 6:7 Mask 8:9 Compare value When Region Type is 10: Word Description 6:7 Mask 8:9 Reserved
Reserved	10:15	Reserved.

3.8.7.2 Section Length

The length of the NVM allowlist section in words.

Note: Section length does not include a count for the section length word.

Bits	Field Name	Value	Description
15:0	Section Length		Section Length Length in words of the section.

3.8.7.3 Number of Accessible CSR Blocks

Number of valid entries in the IOSF-SB CSR Access allowlist.

Bits	Field Name	Value	Description
15:0	Number of Regions		Number of Regions The number of valid entries of accessible regions defined within a given allowlist subsection. The value can range from 0 to 8 MAC CSR allowlist and 0 to 32 in IOSF-SB, I ² C Devices and MDIO Devices allowlists. When the number of defined accessible CSR blocks is 0 the CSR access control is disabled and the DNL engine or the host admin command is provided read/write access to the whole CSR space.

3.8.7.4 Target ID

The Target ID word takes different format on each of the allowlist sub sections. Table 3-235 through Table 3-238 describes the Target ID format for each allowlist section.

Table 3-235. Target ID - MAC CSR Section

Bits	Field Name	Value	Description
15	DNL		DNL Allowed Source This region applies to write transactions originated from Dreadnought Lake (DNL) script.
14	AQC		AQC Allowed Source This region applies to write transactions originated from "Neighbor Device Request" admin command. Note: When I ² C access is monitored, the write transaction can be originated from "Write I2C" or "Read/Write SFF EPROM" admin commands. Note: When MDIO access is monitored, the write transaction can be originated from "Write MDIO" admin command.
13:8	Reserved		Reserved.
7:0	CSR Region		CSR Region Specifies the CSR region, with the following coding: 0 = BAR 0 Note: Currently only BAR0 registers are accessible using DNL scripts.

Table 3-236. Target ID - SB-IOSF Section

Bits	Field Name	Value	Description
15	DNL		DNL Allowed Source This region applies to write transactions originated from Dreadnought Lake (DNL) script.
14	AQC		AQC Allowed Source This region applies to write transactions originated from "Neighbor Device Request" admin command. Note: When I ² C access is monitored, the write transaction can be originated from "Write I2C" or "Read/Write SFF EPROM" admin commands. Note: When MDIO access is monitored, the write transaction can be originated from "Write MDIO" admin command.
13:8	Reserved		Reserved.
7:0	IOSF-SB TID		IOSF-SB Target ID Specifies the target sideband device ID, with the following coding: 0:15 = According to RDA sideband target ID table (Table TBD). 16 = The E610 PHYSS. All other values are reserved.

Table 3-237. Target ID - I²C Section

Bits	Field Name	Value	Description
15	DNL		DNL Allowed Source This region applies to write transactions originated from Dreadnought Lake (DNL) script.
14	AQC		AQC Allowed Source This region applies to write transactions originated from "Neighbor Device Request" admin command. Note: When I ² C access is monitored, the write transaction can be originated from "Write I2C" or "Read/Write SFF EPROM" admin commands. Note: When MDIO access is monitored, the write transaction can be originated from "Write MDIO" admin command.

Table 3-237. Target ID - I²C Section [continued]

Bits	Field Name	Value	Description
13	Reserved		Reserved.
12:10	I2C Bus		I²C Bus The I2C bus to which this region applies.
7:0	I2C Bus Address		I²C Bus Address I ² C device address for 7-bit or 10-bit devices. For 7-bit devices, the 3 high MSB should be set to 0.

Table 3-238. Target ID - MDIO Section

Bits	Field Name	Value	Description
15	DNL		DNL Allowed Source This region applies to write transactions originated from Dreadnought Lake (DNL) script.
14	AQC		AQC Allowed Source This region applies to write transactions originated from "Neighbor Device Request" admin command. Note: When I ² C access is monitored, the write transaction can be originated from "Write I2C" or "Read/Write SFF EPROM" admin commands. Note: When MDIO access is monitored, the write transaction can be originated from "Write MDIO" admin command.
13	Reserved		Reserved.
12:10	MDIO Bus		MDIO Bus The MDIO bus to which this region applies.
9:5	MDIO Bus Address		MDIO Bus Address The MDIO bus address to which this region applies.
4:0	MDIO Device		MDIO Device Number The MDIO device number to which this region applies.

3.8.7.5 Region Control Word

The Region Control word is added to provide indirect access detection and control within the allowlist. Three types of allowlist regions are defined (*Region Type* field).

- **Normal Direct/Indirect Region** — A direct or indirect access register area that is allowed for writing. Such region have a start address and end address. A direct region has *Region ID* of 0.
- **Indirect access control gateway register** — A direct access address that contains registers that are used for indirect access. The content written to this register is examined to identify indirect access write operation.
- **Indirect access address gateway register** — A direct access address that contains registers that are used for indirect access. The content written to this register identify the address that is used in the indirect operation.

In all allowlists sub sections (CSR region, IOSF SB, I²C and MDIO), the second word is defined as "Region Control", with the following structure:

Bits	Field Name	Value	Description
15:5	Reserved		Reserved.
4:2	Region ID		Region ID Identifies the region ID. 0 = Direct access region. 1:7 = Indirect access regions. When <i>Region Type</i> is 00b, it identifies the region ID of the register space to which the allowlist item refers. When <i>Region Type</i> is 10b or 01b, it identifies the region ID that the control gateway opens. <i>Region ID</i> must be unique within a allowlist.
1:0	Region Type		Region Type Identifies the region type. 0 = Normal direct/indirect region. 1 = Indirect access control gateway register. 2 = Indirect access control gateway address. All other values are reserved.

3.8.7.6 CSR Block Start Address [15:0]

The start of each CSR block is defined by CSR Block Start Address High and CSR Block Start Address.

Bits	Field Name	Value	Description
15:0	CSR Block Start Address		CSR Block Start Address [15:0]

3.8.7.7 CSR Block Start Address [31:16]

The start of each CSR block is defined by CSR Block Start Address High and CSR Block Start Address.

Bits	Field Name	Value	Description
15:0	CSR Block Start Address		CSR Block Start Address [31:16]

3.8.7.8 CSR Block Start Address [47:32]

The start of each CSR block is defined by CSR Block Start Address High and CSR Block Start Address.

Bits	Field Name	Value	Description
15:0	CSR Block Start Address		CSR Block Start Address [47:32]

3.8.7.9 CSR Block End Address [15:0]

The end of each CSR block is defined by CSR Block End Address High and CSR Block End Address.

Bits	Field Name	Value	Description
15:0	CSR Block End Address		CSR Block End Address [15:0]

3.8.7.10 CSR Block End Address [31:16]

The end of each CSR block is defined by CSR Block End Address High and CSR Block End Address.

Bits	Field Name	Value	Description
15:0	CSR Block End Address		CSR Block End Address [15:0]

3.8.7.11 CSR Block End Address [47:32]

The end of each CSR block is defined by CSR Block End Address High and CSR Block End Address.

Bits	Field Name	Value	Description
15:0	CSR Block End Address		CSR Block End Address [47:32]

3.8.7.12 Start Address and End Address for Indirect Access Control

When *Region Type* in the Region Control word of the allowlist item is equal to 01b, the allowlist item takes the following structure:

- *Start Address* — This field is used to identify the address of the control gateway register.
- *End Address* — This field is re-defined as follows:
 - Mask: 32 bits to mask the data in order to identify the indirect write operation.
 - Compare: 32 bits to compare the masked data in order to identify the indirect write operation.

When *Region Type* in the Region Control word of the allowlist item is equal to 10b, the allowlist item takes the following structure:

- *Start Address* — This field is used to identify the address of the address gateway register.
- *End Address* — This field is re-defined as follows:
 - Mask: 32 bits to mask the data in order to identify which bits are taken for address.
 - Compare: Reserved.

Note: All the control gateway allowlist items that defines a region ID must appear in the allowlist before the items defining the allowlist regions of the specific indirect region ID.

Chapter 4 Initialization

4.1 E610 Power-Up

4.1.1 POR Flow

Power-up sequence described in this document starts at power-on reset towards the E610.

1. Power-on reset is de-asserted. Crystal clocks are stable and toggling, and memory repair process has completed and the E610 starts early boot stage.
2. Fuse sensing is complete. The IOSF SB network is ready and early boot IPs perform fuse pulling. At that stage, the SPI controller is ready to accept messages.
3. Core PLLs are tuned and getting lock and the E610 perform fuse pulling and ready to start. At this point, global reset is de-asserted towards the E610.
4. The E610 derives an ECDSA384 key-pair from the Unique Device Secret (UDS). The private key is stored in secure memory in the crypto accelerator block, while the public key is exposed to the host in device registers.

5. The E610 starts its boot process. The E610 checks the validity of the NVM and loads the initial firmware code (mini-loader). The mini-loader code is authenticated by E610.

Note: PCI reset de-assertion occurs at least 100 ms after power-up flow starts. Since flow duration is less than 100 ms, it is guaranteed that firmware is ready to PCI reset de-assertion at the end of this stage.

- a. E610 firmware loads a small subset of configurations from the NVM into Shadow RAM (timing critical configurations or ones that might be later modified by software).
6. First load process, E610 firmware loads:
 - a. Pre-load configuration. Autoloaded data is pushed by the E610.
 - b. CSR protected list.
 - c. POR section from Shadow RAM.
7. E610 firmware loads and authenticates the full firmware code content.
8. Second load process:
 - a. E610 firmware loads the full HIP configuration.

Figure 4-1 shows the stages through the power-up sequence. The numbers relate to the steps previously described.

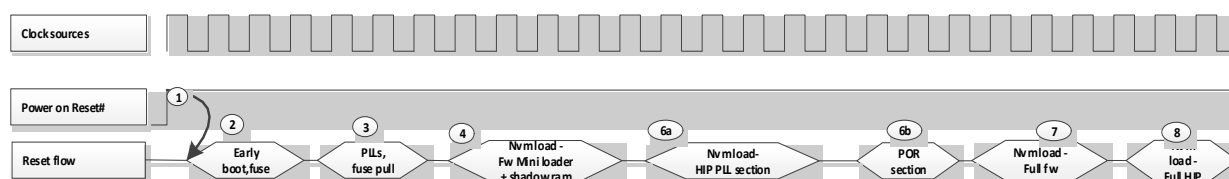


Figure 4-1. Power-On Flow

Figure 4-1 lists the timing of each of the steps.

Table 4-1. Power-Up Sequence Timing

Step		Duration (ms) ¹	Completion Time From Start (ms) ²
1	Power on reset	0	40
2	Read fuse and straps information	5	45
3	PLL lock	45	46
5	Load mini-loader from the NVM ³ + Load shadow RAM from the NVM	25	71
6	Initial configuration of hardware blocks (autoload).		

1. Time it takes to complete the respective step (usually defines the maximum duration).
2. The time the respective step ends, counting from beginning of the power-up sequence. It usually defines the latest time this step might end.
3. All NVM loads assumes an NVM clock speed of 50 MHz and a quad SPI interface + authentication process done by firmware.

4.2 Reset Operation

E610 reset sources are described in the sections that follow.

4.2.1 Reset Sources/Reset Order

The different reset sources are organized in the following order (high-to-low):

9. Power Good Reset
10. P-IOSF reset

A higher-level reset is either followed by a lower-level reset or directly resets the information related with a lower-level reset.

4.2.1.1 Power Good Reset

This reset signal is the main reset. It is level sensitive, and while it is 0b, all of the registers are held in reset.

Power Good Reset changes state during system power up.

4.2.1.2 P-IOSF Reset

The primary reset signal is kept at the low level when the system is at power down state and at system boot. PCIe reset triggers de-assertion of primary reset. If there is no AUX power, the PCI reset generates an internal STRST signal that clears PCIe sticky bits. De-assertion of primary reset also triggers internally a PCIR.

This reset might also reset the internal PHY depending on WoL and manageability requirements.

- **Sticky Reset (STRST)** — Sticky reset is internal signal triggered by PCIe reset when the E610 is not powered by AUX power. This reset clears sticky registers in the PCIe interface (as defined by the PCIe specification). The sticky reset is also initiated by POR.

4.2.1.2.1 P-IOSF Cold/Warm Reset

When the SoC is about to assert reset, it first issues an internal message to enable a graceful reset flow. Upon receiving this message LKP:

- Stops sending new requests out to the PCIe (primary and sideband) interfaces.
- Waits for all pending completion to arrive.
- Responds to all pending non-posted requests.

Further, during the period between receiving this message and the actual reset event, LKP is expected to respond to new incoming requests either normally or with a Unsupported Request (UR).

Once these previous steps complete, LKP sends an acknowledgment to the SoC to indicate that it might perform:

A warm/cold reset is equal to asserting the P-IOSF reset

4.2.1.3 D3hot to D0 Transition

This is also known as an ACPI reset. The E610 generates an internal reset on the transition from D3hot power state to D0 (caused after configuration writes from a D3-to-D0 power state).

Note: This reset is per function and resets only the function that transitions from D3hot-to-D0 and does not reset the configuration space of the function. Its effect is equivalent to a software reset ([Section 4.2.1.5.1](#)).

4.2.1.4 Function Level Reset (FLR) Capability

The *FLR* bit is required for the Physical Function (PF) and per Virtual Function (VF). Setting this bit for a VF only resets the part of the logic dedicated to the specific VF and does not influence the shared part of the port. Setting the PF *FLR* bit resets the entire function.

4.2.1.4.1 FLR in Non-IOV Mode

A FLR reset to a function is equivalent to a D0 → D3 → D0 transition with the exception that this reset does not require any software device driver intervention in order to stop the master transactions of this function. FLR affects the device 1 parallel clock cycle from FLR assertion by default setting, or any other value defined by the *FLR Delay Disable* and *FLR Delay* fields in the PCIe Init Configuration 2 - Offset 0x02 word in the NVM.

4.2.1.4.2 Physical Function FLR (PFLR)

A FLR reset to the PF function in an IOV mode is equivalent to a FLR reset in non-IOV mode. All VFs in the PCIe function of the PF are affected.

The affected VFs are not notified of the reset in advance. The *RSTD* bit in *VFMailbox[n]* is set following the reset (per VF) to indicate to the VFs that a PF FLR took place. Each VF is responsible to probe this bit (such as after a timeout).

4.2.1.4.3 Virtual Function FLR (VFLR)

A VF operating in an IOV mode can issue a FLR. VFLR resets the resources allocated to the VF (like disabling the queues and masking interrupts). It also clears the PCIe configuration for the VF. There is no impact on other VFs or on the PF.

Tx and Rx flows for the queues allocated to this VF are disabled. All pending read requests are dropped and PCIe read completions to this function can be completed as unsupported requests.

Notes:

- Clearing the IOV *Enable* bit in the IOV structure is equivalent to a VFLR to all VFs in the same port.
- A VFLR does not release queues that were blocked due to malicious events. The PF device driver needs to release them using the matching bit in WQBR_RX and WQBR_TX registers. The matching bits in these registers should be cleared after each VFLR, even if not caused due to a malicious event.
- PF driver should clear the VF's VFMBMEM after a VFLR is detected.

4.2.1.5 Soft Resets

4.2.1.5.1 Software Reset

A software reset is done by writing to the Device Reset bit of the Device Control (CTRL.RST) register. The E610 re-reads the per-function NVM fields after software reset. Bits that are not normally read from the NVM are reset to their default hardware values.

Note: This reset is per function and resets only the function that received the software reset.

PCI configuration space (configuration and mapping) of the device is unaffected. The MAC might or might not be reset (see [Section 4.2.4](#)).

Prior to issuing a software reset, the software device driver needs to execute the master disable algorithm as defined in [Section 5.2.4.3.2](#).

If DCB is enabled, then following a software reset using the following steps must be executed to prevent potential races between manageability mapping to TC before and after initialization.

1. Clear the Flow Control enablement in the MAC by clearing the MFLCN.RFCE (or just clear the entire register).
2. Software should wait ~10 μ s.
3. Software polls the TFCS.TC_XON[0] = 0b (most of the time it is expected to be found at 0b while maximum poll time is always shorter than the maximum expected PAUSE time before the software reset was initiated).
4. Software maps the manageability transmit TC (setting the MNGTXMAP register) and then map the user priority of manageability traffic to the manageability TC (setting the RTRUP2TC and RTTUP2TC registers).
5. Software then waits ~10 μ s.
6. Software can re-enable the FC as part of the rest of the initialization flow.

4.2.1.5.2 Physical Function (PF) Software Reset

A software reset by the PF in IOV mode has the same consequences as a software reset in a non-IOV mode.

The procedure for PF software reset is as follows:

1. The PF driver disables master accesses by the device through the master disable mechanism (see [Section 5.2.4.3.2](#)). Master disable affects all VFs traffic.
2. Execute the procedure described in [Section 4.2.2](#) to synchronize between the PF and VFs.

VFs are expected to timeout and check on the *RSTD* bit in order to identify a PF software reset event. The *RSTD* bits are cleared on read.

4.2.1.5.3 VF Software Reset

A software reset applied by a VF is equivalent to a FLR reset to this VF with the exception that the PCIe configuration bits allocated to this function are not reset. It is activated by setting the *VTCTRL.RST* bit.

4.2.1.5.4 Force TCO

This reset is generated when manageability logic is enabled. It is only generated if enabled by the *Force TCO Reset* bit in the Common Firmware Parameters word in the NVM. If enabled by the NVM, the firmware triggers a port reset by setting the *CTRL.RST* bit. In pass-through mode it is generated when receiving a NC-SI command with bit 0 set.

4.2.1.6 Link Reset

Initiated by writing the *Link Reset* bit of the Device Control register (*CTRL.LRST*).

A link reset is equivalent to a software reset + reset of the MAC + reset of the PHY. The E610 rereads the per-function NVM fields after link reset. Bits that are not normally read from the NVM are reset to their default hardware values.

Note: This reset is per function and resets only the function that received the link reset.

The PF in IOV mode can generate a link reset.

Prior to issuing link reset the software device driver needs to execute the master disable algorithm as defined in [Section 5.2.4.3.2](#).

4.2.1.7 PHY Resets

A PHY reset event causes a link down and restarts the auto-negotiation process. This can take a few seconds to complete, which might result in the drop of the TCP sessions with the host and/or with a Manageability Controller (MC). Because the PHY should be accessed by the MC only (via internal firmware), the software device driver should coordinate any PHY reset with the firmware using debug admin command with the following limitation:

Ensure that the *MMNGC.MNG_VETO* bit is cleared. If it is set, the MC requires a stable link and thus the PHY should not be reset at this stage. The software device driver can wait for this bit to be cleared by the MC. See [Section 5.3.5.2](#) for more details on *MNG_VETO* bit.

4.2.2 Reset in a PCI-IOV Environment

Several mechanisms are provided to synchronize reset procedures between the PF and the VFs.

4.2.2.1 RSTI/RSTD

This mechanism is provided specifically for a PF software reset but can be used in other reset cases as follows.

- One of the following reset cases takes place:
 - Power Good Reset
 - P-IOSF Reset
 - D3hot-to-D0 transition
 - FLR
 - Software reset by the PF
- The E610 sets the *RSTI* bits in all the VF Mailbox registers. Once the reset completes, each VF can read its VF Mailbox register to identify a reset in progress.
 - The VF can poll the *RSTI* bit to detect if the PF is in the process of configuring the device.
- Once the PF completes configuring the device, it sets the *CTRL_EXT.PFRSTD* bit. As a result, the E610 clears the *RSTI* bits in all the VF Mailbox registers and sets the *Reset Done (RSTD)* bits in all the VFMailbox registers.
 - The VF might read the *RSTI* bit to detect that a reset has occurred. The *RSTI* bit is cleared on read.

4.2.2.2 VF Receive Enable — PFVFRE / VF Transmit Enable — PFVFTE

This mechanism insures that a VF cannot transmit or receive before the Tx and Rx path have been initialized by the PF.

- The PFVFRE register contains a bit per VF. When the bit is set to 0b, assignment of an Rx packet for the VF's pool is disabled. When set to 1b, the assignment of an Rx packet for the VF's pool is enabled.
- The PFVFTE register contains a bit per VF. When the bit is set to 0b, fetching data for the VF's pool is disabled. When set to 1b, fetching data for the VF's pool is enabled. Fetching descriptors for the VF pool is maintained, up to the limit of the internal descriptor queues — regardless of PFVFTE settings.

PFVFTE and PFVFRE are initialized to zero (VF Tx and Rx traffic gated) following a PF reset. The relevant bits per VF are also initialized by a VF software reset or VFLR.

4.2.3 Firmware Reset

A firmware reset is caused either due to an internal fault in the firmware code, or due to a Force TCO SMB or NC-SI command with bit 2 set.

After a reset the firmware will initiate an interrupt in the EICR.MNG field. The software device driver can detect a firmware reset by an increase in the value of the GL_FWRESETCNT register.

4.2.4 Reset Effects

The resets listed in [Section 4.2.1](#) affect the following registers and logic:

Table 4-2. Reset Effects — Common Resets

Reset Activation	P_IOSF Reset (PERST/In band)	Firmware Reset (EMPR)	Force TCO	Additional Notes
NVM read		X		
PCI configuration registers (RO)	X			7
PCI configuration registers (RW)	X			7
PCIe local registers				
Data path	X		X	1, 6
MAC, TimeSync	X ¹		X	
Integrated PHY (state only reset)	X ¹		X	
Integrated PHY (full autoload)				
Wake up (PM) Context	X ²			2
Wake up/manageability control/status registers				3, 4
Manageability unit		X		
LAN disable strapping pins	X			
All other strapping pins				
Shadow RAMs in MAC or PHY				

1. The MAC cluster and the PHY are reset by the appropriate event only if the manageability unit is disabled and the host is in a low-power state with WoL disabled. WoL disabled means either AUX_PWR pin is cleared, or the *APM Enable* bit in NVM Control Word 3 is disabled, or ACPI is disabled (all wake up filters are disabled or *PME_En* bit is disabled in PMCSR register).
2. If *AUX_PWR* = 0b the wake up context is reset (*PME_Status* and *PME_En* bits should be 0b at reset if the E610 does not support PME from D3cold).

Table 4-3. Reset Effects — Per Function Resets

Reset Activation	D3 or Dr	FLR or PFLR	Software Reset	Exit from LAN Disable	Additional Notes
NVM read					
PCI configuration registers (RO)					7
PCI configuration registers (RW)		X			7, 8
Data path and memory space, TimeSync	X	X	X	X	1, 6
MAC	X ¹	X ²	X ²	X	
Integrated PHY (state only reset)	X ¹	X ²		X	
Virtual function resources	X	X	X		9
Wake up (PM) context					2
Wake up/manageability control/status registers					3, 4
Manageability unit					
Shadow RAM					

1. The MAC cluster and the PHY are reset by the appropriate event only if the manageability unit is disabled and the host is in a low-power state with WoL disabled. WoL disabled means either AUX_PWR pin is cleared, or the *APM Enable* bit in NVM Control Word 3 is disabled, or ACPI is disabled (all wake up filters are disabled or *PME_En* bit is disabled in PMCSR register).
2. Concerned functionalities will not reset in case the Manageability or WoL are enabled.

Table 4-4. Reset Effects — Per Function Resets

Reset Activation	VFLR	VF Software Reset	Additional Notes
Interrupt registers	X	X	10
Queue disable	X	X	11
VF specific PCIe configuration space	X		12
Data path			
Statistics registers			13

Additional notes for Table 4-2 through Table 4-4:

1. The following register fields do not follow the general rules previously described:
 - a. ESDP registers — Reset on Power Good Reset only.
 - b. LED configuration registers — Reset on Power Good Reset and on software reset events.
 - c. The Aux Power Detected bit in the PCIe Device Status register is reset on Power Good Reset and P-IOSF Reset only.
 - d. FLA — Reset on Power Good Reset only.
 - e. RAH/RAL[n, where n>0], MTA[n], VFTA[n], WUPM[n], FFMT[n], FFVT[n], TDBAH/TDBAL, and RDBAH/RDVAL registers have no default value. If the functions associated with these registers are enabled they must be programmed by software. Once programmed, their value is preserved through all resets as long as power is applied.
 - f. Statistic registers (physical function)

2. The wake up context is defined in the PCI Bus Power Management Interface specification (sticky bits). It includes:
 - a. *PME_En* bit of the Power Management Control/Status Register (PMCSR).
 - b. *PME_Status* bit of the PMCSR.
 - c. *Aux_En* bit in the PCIe registers.
 - d. The device requester ID (since it is required for PM_PME TLP).The shadow copies of these bits in the Wake Up Control (WUC) register are treated identically.
3. Refers to bits in the WUC register that are not part of the wake up context (the *PME_En* and *PME_Status* bits). The WUFC register is not part of the wake up context and is reset as part of the data path.
4. The Wake Up Status (WUS) registers include the following:
 - a. WUS register.
 - b. Wake Up Packet Length (WUPL) register.
 - c. Wake Up Packet Memory (WUPM) register.
5. The MAC cluster and the PHY are reset by the appropriate event only if the manageability unit is disabled and the host is in a low-power state with WoL disabled. WoL disabled means either *AUX_PWR* pin is cleared, or the *APM Enable* bit in NVM Control Word 3 is disabled, or ACPI is disabled (all wake up filters are disabled or *PME_En* bit is disabled in PMCSR register).
6. The contents of the following memories are cleared to support the requirements of PCIe FLR:
 - a. The Tx packet buffers.
 - b. The Rx packet buffers.
7. Sticky bits and hardware initialization bits (indicated as HwInit) in the PCI Configuration registers are cleared only by Power Good Reset (Global Reset).
8. The following register fields are not affected by FLR or PFLR:
 - a. *Max_Payload_Size* in the Device Control register.
9. These registers include:
 - a. VFEICS
 - b. VFEIMS
 - c. VFEIAC
 - d. VFEIAM
 - e. VFEITR 0-2
 - f. VFIVAR0
 - g. VFIVAR_MISC
 - h. VFPBACL
 - i. VFMailbox

10. These registers include:

- a. VFEICS
- b. VFEIMS
- c. VFEIMC
- d. VFEIAC
- e. VFEIAM
- f. VFEICR
- g. EITR 0-2
- h. VFIVAR0
- i. VFIVAR_MISC
- j. VFPBACL
- k. VFMailbox
- l. RSCINT

11. These registers include specific VF bits in the FVRE and FVTE registers are cleared as well.

12. These registers include:

- a. MSI/MSI-X enable bits
- b. BME
- c. Error indications

13. Rx and Tx counters might miss proper counting due to VFLR indicating more packets than those ones actually transferred. It could happen if VFLR happened after counting occurred but before Tx or Rx completed.

Note: Unless specified otherwise the E610's on-die memories are reset together with the functional block(s) they belong to.

4.2.5 Autoload Shadow RAM

This phase of autoload determines if a properly-configured Flash device is attached:

- If the attached Flash device is not properly configured, the default hardware settings are kept. The power-on flow continues without loading from the NVM.
 - The NVM is expected to be reprogrammed. See [Section 3.3.4.2](#) for more details on how this is done.
- If the attached Flash device is properly configured, autoload proceeds with its first stage. NVM modules loaded to the on-die shadow RAM are read from the NVM.

Once the shadow RAM is ready, an internal indication is set that the device enumeration may proceed. This indication is also set when the NVM is found blank.

4.2.5.1 Autoload into Device Units

The NVM configurations are loaded into the device units directly from the NVM or indirectly through shadow RAM.

The loading process of the NVM modules that are specified at table at [Table 4-2](#) at [Section 4.2.4](#) are tracked and reported.

The following CSR fields track the progress of loading the NVM modules following power on and the various resets:

- GLNVM_ULD.*_DONE bits (one per module) indicates the unit is read.

Default hardware values:

- GLNVM_ULD.*_DONE = 0b
- EEC*.*_DONE = 0b

The flow during a power-on stage is as follows.

- If the NVM is found blank:
 - All GLNVM_ULD.*_DONE and EEC*.*_DONE bits are set by the E610
- If the NVM is found valid:
 - After process is done by EMP firmware it sets the GLNVM_ULD.*_DONE and EEC*.*_DONE bits to 1b.

4.2.5.2 Firmware Initialization

Once EMP firmware loads, the following E610 features are enabled:

- The external Ethernet link is active
- Communication with the BMC is possible if supported by the E610 and the system (optional)
- OEM specific manageability agents are active and responding to commands from the Ethernet fabric (optional)
- Admin queues CSR based for enabled PCI functions are ready for commands - EMP responds to each function's Get Version AQ command to indicate that it is safe for software to start using the E610 for device driver initialization.

Once EMP firmware is up and running, the E610 can be configured via its management interfaces, and certain device capabilities are then enabled or disabled (such as soft SKUing).

4.2.5.3 MAC Address Initialization

The E610 supports a port MAC Address for each of its ports. These addresses are used for layer 2 flow control pause frames and WoL purposes. The addresses are initially loaded from the NVM into the RAH/RAL[0] registers and updated in the initialization flow according to the below table.

Table 4-5 lists how each address can be set or read and where it is stored:

Table 4-5. MAC Address Initialization

MAC Address Type	Where Stored	How Reported	How Modified	Modified at Manufacturing?
LAN MAC Address - factory	NVM PF MAC Address section (TLV type = 0x10F) Loaded by firmware to: • RAL/RAH[0]	Dell Get Address NC-SI Command (Permanent MAC Address)	N/A	Yes
LAN MAC Address - BIOS setting	Alternate RAM (Current LAN MAC Address (Low/High)) Loaded by firmware to: • RAL/RAH[0]	Dell Get Address NC-SI Command (Virtual MAC Address) Manage MAC Address read (LAN address valid) - if no LAA	Write alternate AQ command: • Alternate LAN MAC Address (LS) • Alternate LAN MAC Address (MS) • NC-SI Set Address OEM command	No
WoL MAC Address	RAL/RAH[0]	Manage MAC Address read (WoL address valid)	direct access from PF	Yes
MNG MAC Address	MNG MAC Address NVM Module	NVM read command (module type = 0x110)	NVM write command (module type = 0x110)	No
PCIe Serial Address	PCIe Serial address NVM Module	NVM read command (module type = 0x133)	NVM write command (module type = 0x133)	Yes

Notes:

- The AV field in RAH/RAL is set by the EMP when writing a MAC Address.
- The per-PF MAC Address used by the internal switch is managed via the Remove MAC, VLAN pair and Add MAC, VLAN pair commands.

4.3 Clocking

The E610 package contains E610 SoC as well as the PHY 10GBASE-T component. The E610 package will have the capability to support both on board reference clocking as well as on-die reference clocking for E610 SoC. The following sections describe the reference clocking strategy for E610 as well as 10GBASE-T component.

4.3.1 Die Level Clock Interfaces

Please refer to the IO chapter for details on the platform clock interfaces. [Table 4-6](#) shows the usage of the GPIO pin strap which helps to set the static configuration for the E610 SoC reference clock.

Table 4-6. GPIO Pin Strap Usage for E610 SoC Reference Clock

SDP23 (xtal_refclk_sel_n)	Selected Source	Description
0 (Default)	xtal_ref	SDP23 drives E610 SoC reference clock and AON clock selection. 0 (Default): On board 25MHz crystal 1: N/A
1	refa_dig_fr_clk	

[Table 4-7](#) shows the usage of the GPIO pin strap which helps to set the static configuration for the E610 Ethernet PHY reference clock.

Table 4-7. GPIO Pin Strap Usage for E610 Ethernet PHY Reference Clock

SDP22 (eth_refclk_sel)	Selected Source	Description
0	ref_alt0_clk_p/m	SDP22 drives ref_repeat_clk_sel[0] to select source of ETH PHY reference clock and ref_repeat_clk_p/n; 0: On die PLL generated 156.25 MHz. 1: (Default): On board 156.25 MHz oscillator.
1 (Default)	ETH_REFCLK_P/N	

[Table 4-8](#) gives a quick overview of the platform electrical requirements for the E610 and BASE-T components. For details on each of the electrical requirements for each of the I/Os, refer to [Chapter 11, "Electrical/Mechanical Specification"](#).

Table 4-8. Platform Electrical Requirements

Components	Details
E610 Die	Ethernet SerDes @156.25 MHz from pad or from on-die PLL. 1G Usage – 4.5 ps in 12 KHz to 20 MHz Max RMS jitter. 10G Usage – 1 ps Max RMS jitter. SoC PLL(s) @156.25 MHz (Ethernet reference clock) or @25 MHz (crystal).
1GBASE-T Component	XTAL reference 25 MHz +/- 50 ppm at 4.5 ps max RMS jitter (12 KHz to 20 MHz).
10GBASE-T Component	XTAL reference 25 MHz/156.25 MHz +/- 50 ppm at 1 ps Max RMS jitter (80 Hz to 20 MHz). Platform Ask: May use Ethernet 156.25 MHz oscillator. Need to support both dual or quad port designs.

4.3.2 E610 Clocking Requirements

These sections outline the E610 SoC clocking requirements.

4.3.2.1 Core Clocking Support

Table 4-9. Core Clocking Support

Trunk	Source	Description	Top Frequency (MHz)	Other Supported Frequencies (MHz)	AON Clock Prior to PLL lock	Source Power Domain
io_xtal_25_clk	XTAL/PHYSS	Always-on clock	25	39.0625	VDD (0.75V) io_xtal_25_clk	VDD (0.75V)
pll1_car_156p25_clk	PLL1	PHY, IOSF SB, PCS - MAC (5G)	156.25	N/A	io_xtal_25_clk	VDD (0.75V)
pll1_car_312p5_clk	PLL1	MAC, MNG, DBU, PCS-MAC (10G)	312.5	N/A	io_xtal_25_clk	VDD (0.75V)
pll1_car_500_clk	PLL1	IOSF P	500	N/A	io_xtal_25_clk	VDD (0.75V)
pll1_car_125_clk	PLL1	SPI controller	125	N/A	io_xtal_25_clk	VDD (0.75V)
tap_clk	GPIO	JTAG TCK	100	N/A	N/A	VDD (0.75V)

4.3.2.2 Ethernet PHY Reference Clocking Support

Table 4-10. Ethernet PHY Reference Clocking Support

Trunk	Source	Description	Top Frequency (MHz)	Other Supported Frequencies (MHz)	AON Clock Prior to PLL Lock	Source Power Domain
eth_phy_refclk_p/m	PLL2	Ethernet SerDes Reference Clock	156.25	N/A	N/A	VP (0.9V)

4.3.3 Clocking Bring-Up

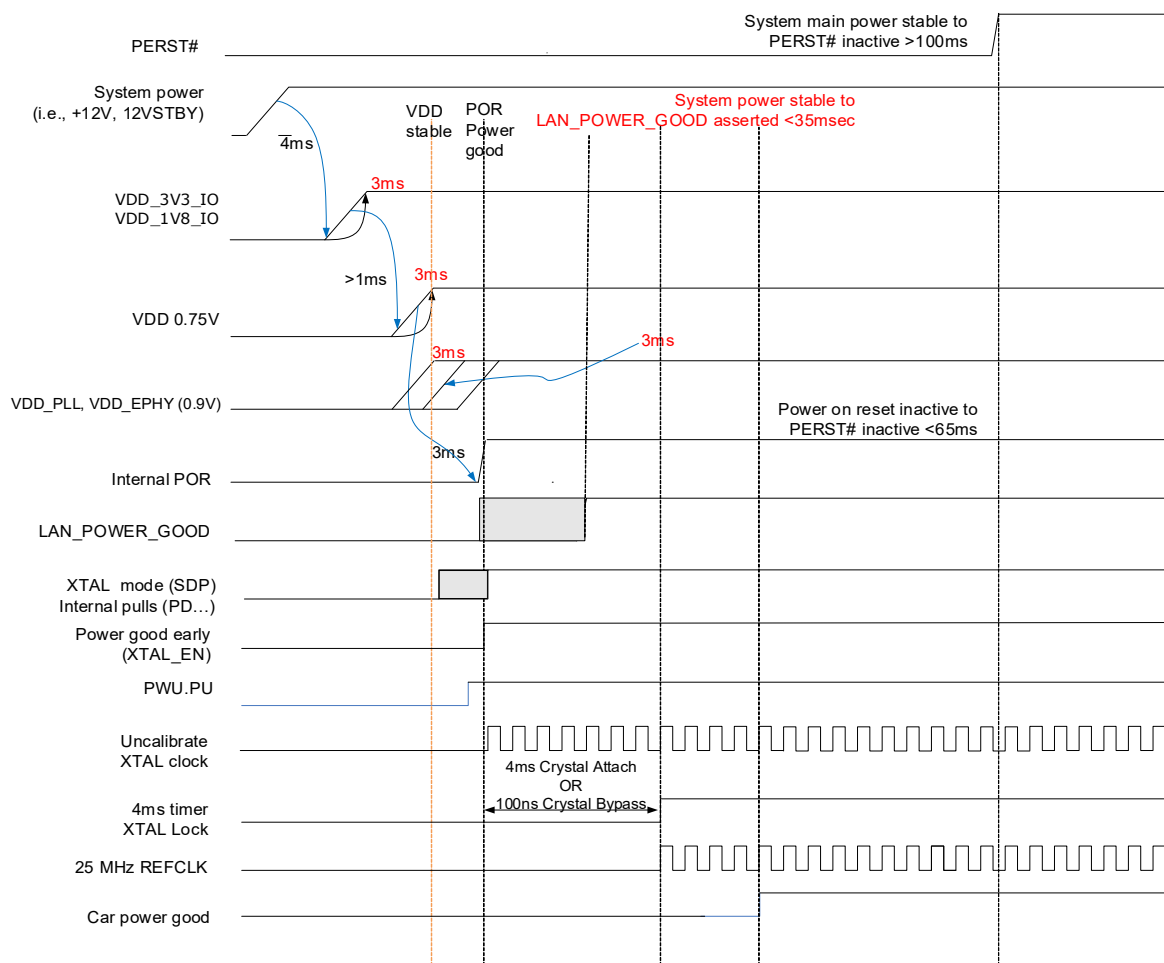


Figure 4-2. Clocking Bring-Up

4.3.4 Clocking Shutdown

The clocking infrastructure will shut down during the following conditions

- LAN_PWR_GOOD or POR_PWR_GOOD is dropped resulting a hardware shutdown on E610.
- A catastrophic event such as thermal trip will also shutdown E610 however this is managed via firmware.

4.4 Queue Disable

See [Section 4.6.7.1.1](#) for details on disabling and enabling an Rx queue.

See [Section 4.6.8.1.1](#) for details on disabling and enabling a Tx queue.

4.5 Device/Port/Function Configuration

4.5.1 General

The E610 provides several mechanisms to configure which PCI functions and Ethernet ports are exposed.

The functions and ports can be hidden or exposed:

- Through NVM configuration
- Based on power management policy
- Through strapping pins

4.5.2 Disable Through Strapping Pins

For a LAN on Motherboard (LOM) design, it might be desirable for the system to provide BIOS setup capability for selectively enabling or disabling E610 PCI devices, PCI functions, or external Ethernet ports and the associated PCI function. It provides end users more control over system resource management and avoids conflicts with add-in NIC solutions. The E610 provides support for selectively enabling or disabling one or more LAN PCI device(s) in the system.

The E610 provides options to disable its external Ethernet ports and/or PCI functions:

- A strapping pin (DEV_DIS_N) is sampled on RSMRST# and PCIe resets to disable Ethernet ports. The specific port(s) to be disabled is determined from NVM. Additional NVM configuration is provided to determine which PCI functions are disabled at the same time. The expected usage is to disable the PCI functions associated with the disabled ports.
- A strapping pin (PCI_DIS_N) is sampled on RSMRST# and PCIe resets to disable PCI functions. The specific functions that are disabled are determined from the NVM.

Some guidelines on usage of these straps:

- During power-up, the PCI_DIS_N and DEV_DIS_N pins are ignored until the NVM is read. From that point, the E610 might disable all PCI functions if either PCI_DIS_N or DEV_DIS_N is asserted.
- De-assertion of the PCI_DIS_N or DEV_DIS_N pins requires the system to issue a power-on reset/ RSMRST#/PE_RST_N to the E610 in order to re-enable any disabled PCI functions or external Ethernet ports.

- The PCI_DIS_N and DEV_DIS_N pins should maintain their state during system reset and system sleep states. It should also insure the proper default value on system power up. For example, one could use a GPIO pin that defaults to 1b (enable) and is on system suspend power (such that it maintains its state in S0-S5 ACPI states).

When function 0 is disabled then it does not disappear from the PCIe configuration space. Rather, the function presents itself as a dummy function. The device ID and class code of this function changes to other values (dummy function device ID 0x10A6, class code 0xFF0000). In addition, the function does not require any I/O space, and does not require an interrupt line. It requires a minimal memory space (4 KB) that is not mapped to internal registers.

Mapping between function and LAN ports is listed in the following table.

Table 4-11. PCI Functions Mapping

PCI Function #	LAN Function Select	Function 0	Function 1
Both LAN functions are enabled.	0	LAN 0	LAN 1
	1	LAN 1	LAN 0
Function 0 is disabled.	0	Dummy	LAN 1
	1	LAN 1	Disable
Function 1 is disabled.	0	LAN 0	Disable
	1	Dummy	LAN 0
Both LAN functions are disabled.	Both PCI functions are disabled. Device is in low-power mode.		

The following rules apply to function disable:

- When function 0 is disabled, it is converted into a dummy PCI function. Function 1 is not affected.
- When function 1 is disabled, it disappears from the PCI configuration space.
- The disabled LAN port is still available for manageability purposes through the NVM mechanism.
- The disabled LAN port is still available for manageability purposes if disabled through the NVM. In this case, and if the *LPLU* bit is set, the PHY firmware attempts to create a link at the lowest supported speed. The disabled LAN port is not available for manageability purposes if disabled through the strapping mechanism.
- Dummy function mode should not be used in PCI IOV mode (since PF0 is required to support certain functionality).

The following NVM bit control function disable:

- The LAN_FUNCTION_SEL field in NVM (reflected in FACTPS.LAN_FUNCTION_SEL bit) defines the correspondence between LAN port and PCI function.

When a particular LAN is fully disabled (Ethernet port with associated PF), all internal clocks to that LAN are disabled, the device is held in reset, and the internal PHY for that LAN is powered down. In both modes, the device does not respond to PCI configuration cycles. Effectively, the LAN device becomes invisible to the system from both a configuration and power consumption standpoint.

4.5.3 Port and Device Disable

The following mechanisms are provided to enable and disable Ethernet ports:

- NVM configuration:
 - Ports are enabled or disabled from the NVM. The `PRTGEN_CNF.PORT_DIS` per-port register bit (loaded from the NVM) disables external ports other than port 0 (`PORT_DIS` for port 0 is hard-wired to always enabled).
 - When cleared, the respective port is enabled.
 - The hardware default value is for port 0 to be enabled and other ports to be disabled.
- Power management:
 - If the Ethernet ports are not required in Dr state (such as if manageability is disabled and APME WoL is disabled as well), then all ports are disabled during Dr state.
- Strapping (`DEV_DIS_N`):
 - When the `DEV_DIS_N` pin is asserted low, the `PRTGEN_CNF.ALLOW_PORT_DIS` per-port bits (loaded from the NVM) determine if the port is disabled.
 - The hardware default value for these bits is 0x0 (do not disable).
 - If a bit is set to 0b, `DEV_DIS_N` has no effect on the port.

Note: If a port is required for manageability purposes, it should not be disabled by the mechanisms previously described.

The result of the previous mechanisms is reflected in the `FACTPS.PORT_VALID` bit (per port), which denotes if the port is enabled. A bit is cleared (port disabled) if at least one of the previous mechanisms disables the port. The port is then powered down (including MAC, PCS, PHY).

If all ports are disabled, the E610 is put in a power-down, reset state. Specifically, the E610 does not respond to PCI configuration cycles, the PCIe link is in L3 state, and Ethernet ports are in power down. All PCI functions must be disabled as well via the mechanisms previously described (such as the proper NVM configuration must be set to disable for PFs).

4.5.4 Function Disable

The following mechanisms are provided to enable and disable PCI functions.

- NVM configuration:
 - PCI functions are enabled or disabled from the NVM. The `PCI_FUNC.FUNC_DIS` per-PF bit (loaded from the NVM) disables PCI functions (other than function 0. `FUNC_DIS` for function 0 is hardwired to enabled).
 - When cleared, the respective function is enabled.
 - The hardware default value is for function 0 to be enabled and other functions to be disabled.
- Strapping through `PCI_DIS_N`:
 - When the `PCI_DIS_N` pin is asserted low, the `PFPCI_FUNC.ALLOW_FUNC_DIS` per-PF register bit (loaded from the NVM) determines if the function is disabled.
 - If this bit is set to 0b, `PCI_DIS_N` has no effect on the PCI function.
 - The hardware default value is 0b (keep enabled).

- Strapping through DEV_DIS_N:
 - When the DEV_DIS_N pin is asserted low, the PFPCI_FUNC.DIS_FUNC_ON_PORT_DIS per-PF bit (loaded from the NVM) determines if the function is disabled.
 - If this bit is set to 0b, DEV_DIS_N has no effect on the PCI function.
 - The hardware default value is 0 (do not disable).
- Soft SKU commands:
 - The soft SKU commands are executed either before or after PERST# was de-asserted and in any case, before the first access by BIOS to the E610 (such as before the first PCIe configuration cycle to the E610).
 - When a LAN port is enabled or disabled via a Soft SKU command, the EMP identifies (through the FACTPS.LAN_function_sel) the PCI functions associated with the port.
 - EMP then updates the configuration of the functions enabled or disabled by writing to the PCI_FUNC.FUNC_DIS register bits.
 - PCI_FUNC.FUNC_DIS for function 0 is hardwired to enabled, so is not affected by this mechanism

4.5.5 Port Enable and Disable from BIOS (HII)

The E610 allows disabling a port and its corresponding function as a BIOS setup option (e.g. using HII). The operation involves setting adaptive NVM feature, triggering shadow RAM dump to make this configuration persistent and request PCIe reset, so the new configuration would take effect on enumeration.

Table 4-12 shows the defined set of adaptive NVM features.

Table 4-12. HII Port Disable of Physical Function X

Adaptive NVM feature	Description
HII Port Disable of Physical Function	2 feature IDs, one for each PF. Two configuration IDs: "Enable" – Set "HII port disable by function" bit for function X to 0. "Disable" – Set "HII port disable by function" bit for function X to 1.

The adaptive NVM features controls the following bit in NVM PFA area at a new TLV 0x139, as detailed in Table 4-13.

Table 4-13. HII Port Disable by Function

Field Name	Description
HII Port Disable by Function	One bit per PF. When the bit is set, the firmware should disable the port that is associated with the PF according to FACTPS. The disable of the port should be done using the relevant EMP_PORT_DIS field of the PRTGEN_CNF register.

Table 4-14. Allow HII Port Disable

Field Name	Description
Allow HII Port Disable	Determines whether the present HII configuration feature is present in the configuration menu. If this bit is set (1), the HII configuration feature shall be available; if the bit is clear (0), the HII configuration shall not be available.

4.5.6 Event Flow for Enable/Disable Functions

This section describes the expected flow to disable or enable PCI functions or Ethernet ports. Following a power-on reset/RSMRST#/PE_RST_N/in-band reset, the DEV_DIS_N and PCI_DIS_N signals should be driven high (or left unconnected) for normal operation.

The following example assumes that PCI functions and/or Ethernet ports are being disabled during the BIOS initialization phase:

1. Following a power-up sequence, the DEV_DIS_N and PCI_DIS_N signals are driven high.
2. The PCIe link is established following PE_RST_N.
3. BIOS goes through PCI bus enumeration.
4. BIOS recognizes that the PCI functions in the E610 should be disabled.
5. The BIOS drives the DEV_DIS_N or PCI_DIS_N signal to a low level.
6. The BIOS asserts PCIe reset, either in-band or via PE_RST_N.
7. As a result, the E610 samples the DEV_DIS_N and PCI_DIS_N signals and disables the PCI functions and/or external Ethernet ports.
8. BIOS might do device enumeration a second time (the disabled PCI functions are invisible or changed to dummy function).
9. Proceed with normal operation.
10. Re-enable could be done by driving the DEV_DIS_N and PCI_DIS_N signals high and re-issuing a power-on reset/RSMRST#/PE_RST_N/in-band reset.

4.5.6.1 Multi-Function Advertisement

If one of the LAN devices is disabled and function 0 is the only active function, the E610 is no longer a multi-function device. The E610 normally reports 0x80 in the *PCI Configuration Header* field (*Header Type*), indicating multi-function capability. However, if a LAN is disabled and only Function 0 is active, the E610 reports 0x0 in this field to signify single-function capability.

4.5.6.2 Interrupt Use

When both LAN devices are enabled, the E610 uses the PCI legacy interrupts of both ports for interrupt reporting. The NVM configuration controls the *Interrupt Pin* field of the PCI configuration header to be advertised for each LAN device to comply with PCI specification requirements.

However, if either LAN device is disabled, then the legacy PCI interrupt of Port A must be used for the remaining LAN device, therefore the NVM configuration must be set accordingly. Under these circumstances, the *Interrupt Pin* field of the PCI header always reports a value of 0x1, indicating INTA# pin usage, which means legacy PCI interrupt of Port A is used.

4.5.6.3 Power Reporting

When both LAN devices are enabled, the PCI Power Management register block has the capability of reporting a common power value. The common power value is reflected in the *Data* field of the PCI Power Management registers. The value reported as common power is specified via an NVM field, and is reflected in the *Data* field each time the *Data_Select* field has a value of 0x8.

When only one LAN port is enabled and the E610 appears as a single-function device, the common power value, if selected, reports 0x0 (undefined value), as common power is undefined for a single-function device.

4.6 Software Initialization and Diagnostics

4.6.1 Introduction

This section discusses general software notes for the E610, especially initialization steps. These include:

- General hardware power-up state
- Basic device configuration
- Interrupts initialization
- Initialization of transmit
- Receive initialization
- Link configuration
- Software reset capability
- Statistics
- Diagnostic hints
- Virtualization support initialization
- DCB configuration

4.6.2 Power-Up State

When the E610 powers up, it automatically reads the NVM. The NVM contains sufficient information to bring the link up and configure the E610 for manageability and/or APM wake up. However, software initialization is required for normal operation.

4.6.3 Initialization Sequence

The following sequence of commands is typically issued to the device by the software device driver in order to initialize the E610 to normal operation. The major initialization steps are:

1. Disable interrupts.
2. Issue a Power Good Reset and perform general configuration — see [Section 4.6.3.2](#).
3. Wait for the NVM auto-read completion.
4. Wait for manageability configuration done indication (GLNVM_ULD*.*DONE* o).
5. Wait until the DMA initialization completes (RDRXCTL.DMAIDONE).
6. Setup the PHY and the link.
7. Initialize all statistical counters — see [Section 4.6.5](#).
8. Initialize receive — see [Section 4.6.7](#).
9. Initialize transmit — see [Section 4.6.8](#).
10. Initialize Virtualization support — see [Section 4.6.9](#)
11. Configure DCB — see [Section 4.6.10](#).
12. Enable interrupts — see [Section 4.6.3.1](#).

4.6.3.1 Interrupts During Initialization

Most software device drivers disable interrupts during initialization to prevent re-entrance. Interrupts are disabled by writing to the EIMC register. Note that the interrupts need to also be disabled after issuing a Power Good Reset reset, so a typical driver initialization flow is:

1. Disable interrupts.
2. Issue a Power Good Reset reset.
3. Disable interrupts (again).

After initialization completes, a typical software device driver enables the desired interrupts by writing to the IMS register.

4.6.3.2 Global Reset and General Configuration

Global reset = software reset + link reset.

Device initialization typically starts with a software reset that puts the device into a known state and enables the device driver to continue the initialization sequence. Following a global reset the software driver should poll the CTRL.RST until it is cleared and then wait at least 10 ms to enable a smooth initialization flow.

To enable flow control, program the FCTTV, FCRTL, FCRTN, FCRTV and FCCFG registers. If flow control is not enabled, these registers should be written with 0x0. If Tx flow control is enabled, then Tx CRC by hardware should be enabled as well (HLREG0.TXCRCEN = 1b). Refer to [Section 3.6.5.3.2](#) through [Section 3.6.5.3.5](#) for recommended settings of the Rx packet buffer sizes and flow control thresholds.

Note: FCRTN[n].RTH fields must be set by default regardless if flow control is enabled or not. FCRTN[n].FCEN should be set to 0b if flow control is not enabled as all the other registers previously indicated.

The link inter-connect configuration according to the electrical specification of the relevant electrical interface should be set prior to the link setup. This configuration is done through the PHY image section of the NVM by applying the appropriate settings to the link interconnect block.

4.6.4 Link Initialization

TBD

4.6.5 Initialization of Statistics

Statistics registers are hardware-initialized to values as detailed in each particular register's description. The initialization of these registers begins upon transition to a D0 active power state (when internal registers become accessible, as enabled by setting the *Memory Access Enable* field of the PCIe Command register), and is guaranteed to complete within 1 ms of this transition. Note that access to statistics registers prior to this interval might return indeterminate values.

All statistical counters are cleared on read and a typical software device driver reads them (thus making them zero) as a part of the initialization sequence.

Queue counters are mapped using the RQSMR registers for Rx queues, and TQSM registers for Tx queues. Refer to the RQSMR register section for RQSMR setup, and to the TQSM register section for TQSM setup. Note that if software requires the queue counters, the RQSMR and TQSM registers must be reprogrammed following a device reset.

4.6.6 Interrupt Initialization

4.6.6.1 Working with Legacy or MSI Interrupts

- Software device drivers associate between Tx and Rx interrupt causes and the EICR register by setting the IVAR[n] registers.
- Program the SRRCTL[n].RDMTS per receive queue if software uses the Receive Descriptor Minimum Threshold Interrupt (RDMTI).
- All interrupts should be set to zero — no auto clear in the EIAC register. Following an interrupt, software can read the EICR register to check for the interrupt causes.
- Set the auto mask in the EIAM register according to the preferred mode of operation.
- Set the interrupt throttling in the EITR[n] and GPIE according to the preferred mode of operation.
- Software enables the required interrupt causes by setting the EIMS register.

4.6.6.2 Operating with MSI-X

- The operating system/BIOS sets hardware to MSI-X mode and programs the MSI-X table as part of the device enumeration procedure.
- The software device driver associates between interrupt causes and MSI-X vectors and the throttling timers EITR[n] by programming the IVAR[n] and IVAR_MISC registers.

- Program the `SRRCTL[n].RDMTS` (per receive queue) if software uses the receive descriptor minimum threshold interrupt.
- The `EIAC[n]` registers should be set to auto clear for transmit and receive interrupt causes (for best performance). The `EIAC` bits that control the other and TCP timer interrupt causes should be set to 0b — no auto-clear.
- Set auto mask in the `EIAM` and `EIAM[n]` registers according to the preferred mode of operation.
- Set the interrupt throttling in the `EITR[n]` and `GPIE` registers according to the preferred mode of operation.
- Software enables the required interrupt causes by setting the `EIMS[n]` registers.

4.6.7 Receive Initialization

Initialize the following register tables before receive and transmit is enabled:

- Set `CTRL_EXT.Extended_VLAN` bit if needed.
- Receive Address (`RAL[n]` and `RAH[n]`) for used addresses and Receive Address High — `RAH[n].VAL=0b` for unused addresses.
- Unicast Table Array — `PFUTA`
- VLAN Filter Table Array — `VFTA[n]`
- VLAN Pool Filter — `PFVLVF[n]`
- MAC Pool Select Array — `MPSAR[n]`
- VLAN Pool Filter Bitmap — `PFVLVFB[n]`

Program the Receive Address register(s) (`RAL[n]`, `RAH[n]`) per the station address. This can come from the NVM or from any other means (for example, it could be stored anywhere in the NVM or even in the platform PROM for a LOM design).

Set up the Multicast Table Array (MTA) registers. Assuming the entire table was zeroed by the last reset, only the desired multicast addresses should be permitted (by writing 0x1 to the corresponding bit location). Set the `MCSTCTRL.MFE` bit if multicast filtering is required.

Set up the VLAN Filter Table Array (VFTA) if VLAN support is required. Assuming the entire table was zeroed by the last reset, only the desired VLAN addresses should be permitted (by writing 0x1 to the corresponding bit location). Set the `VLNCTRL.VFE` bit if VLAN filtering is required.

Initialize the flexible filters 0...7 — Flexible Host Filter Table (`FHFT_FILTER`) registers.

Program the different Rx filters and Rx offloads via registers `FCTRL`, `VLNCTRL`, `MCSTCTRL`, `RXCSUM`, `RQTC`, `RFCTL`, `MPSAR`, `RSSRK`, `RETA`, `SAQF`, `DAQF`, `SDPQF`, `FTQF`, `SYNQF`, `ETQF`, `ETQS`, `RDRXCTL`, and `RSCDBU`.

Note: As NFS detection is not supported, the `RFCTL.NFSW_DIS` and `RFCTL.NFSR_DIS` bits should be set to 1b.

Program `RXPBSIZE`, `MRQC`, `PFQDE`, `RTRUP2TC`, `MFLCN.RPFCE`, `MFLCN.RPFCM`, and `MFLCN.RFCE` according to the DCB and virtualization modes. Refer to [Section 4.6.10.3](#).

Enable receive jumbo frames by setting `HLREG0.JUMBOEN` in the following case:

- Jumbo packets are expected. Set `MAXFRS.MFS` to the expected maximum packet size.

Enable receive coalescing if required as described in [Section 4.6.7.2](#).

4.6.7.1 Receive Queues Enable

The following should be done for each receive queue:

1. Allocate a region of memory for the receive descriptor list.
2. Receive buffers of appropriate size should be allocated and pointers to these buffers should be stored in the descriptor ring.
3. Program the descriptor base address with the address of the region (registers RDBAH, RDBAL).
4. Set the length register to the size of the descriptor ring (register RDLEN).
5. Program SRRCTL associated with this queue according to the size of the buffers and the required header control.
6. Set RXDCTL[n].RLPML field enabled by the RXDCTL[n].RLPML_EN limiting the maximum Rx packet size. This setting is optional enabling the software to use smaller buffers than the size defined by the SRRCTL[n].BSIZEPACKET. Software may not use smaller buffers than defined by the SRRCTL[n] on Rx queues that enables RSC.
7. If header split is required for this queue, program the appropriate PSRTYPE for the appropriate headers.
8. Program RXDCTL with appropriate values including the queue *Enable* bit. Note that packets directed to a disabled queue are dropped.
9. Poll the RXDCTL register until the *Enable* bit is set. The tail should not be bumped before this bit was read as 1b.
10. Bump the tail pointer (RDT) to enable descriptors fetching by setting it to the ring length minus one.

Enable the receive path by setting RXCTRL.RXEN. This should be done only after all other settings are done. If software uses the receive descriptor minimum threshold interrupt, that value should be set.

4.6.7.1.1 Dynamic Enabling and Disabling of Receive Queues

Receive queues can be enabled or disabled dynamically using the following procedure.

4.6.7.1.1.1 Enabling

Follow the per-queue initialization described in the previous section.

4.6.7.1.1.2 Disabling

- Disable the routing of packets to this queue by re-configuring the Rx filters.
- Disable the queue by clearing the RXDCTL.ENABLE bit. The E610 stops fetching and writing back descriptors from this queue. Any further packet that is directed to this queue is dropped. If a packet is being processed, the E610 completes the current buffer write. If the packet spreads over more than one data buffer, all subsequent buffers are not written.
- the E610 clears the RXDCTL.ENABLE bit only after all pending memory accesses to the descriptor ring are done. The driver should poll this bit before releasing the memory allocated to this queue.
- Once the RXDCTL.ENABLE bit is cleared the driver should wait an additional amount of time (~100 μ s) before releasing the memory allocated to this queue.

The Rx path can be disabled only after all the receive queues are disabled.

Note: As there could be additional packets in the receive packet buffer targeted to the disabled queue and the arbitration could be such that it would take a long time to drain these packets, if software re-enables a queue before all packets to that queue were drained, the enabled queue could potentially get packets directed to the old configuration of the queue. For example, VM goes down and a different VM gets the queue. The software device driver should delay the re-enablement of the queue until it is guaranteed there are no more packets directed to this queue in the packet buffer.

4.6.7.2 Flow Director initialization

Flow Director initialization is described in [Section 7.1.3.5.11](#).

4.6.8 Transmit Initialization

- Program the HLREG0 register according to the MAC behavior needed.
- Program the TCP segmentation parameters via registers DMATXCTL (while keeping the TE bit cleared), DTXTCPFLGL, DTXTCPFLGH, and TPH parameters via TPH_TXCTRL.
- Refer to the TIPG description in [Section 3.6.6](#) for more details.
- Set RTTDCS.ARBDIS to 1.
- Program the DTXMXSZRQ, TXPBSIZE, TXPBTHRESH, MTQC, and MNGTXMAP registers according to the DCB and virtualization modes. Refer to [Section 4.6.10.3](#).
- Clear RTTDCS.ARBDIS to 0.
- Legacy drivers that uses queue zero and assumes it is enabled by default, should at this stage, set queue zero parameters as described below.
- Enable the transmit path by setting the DMATXCTL.TE bit.

4.6.8.1 Transmit Queues Enable

The following steps should be done once for each transmit queue:

1. Allocate a region of memory for the transmit descriptor list.
2. Program the descriptor base address with the address of the region (TDBAL and TDBAH).
3. Set the length register to the size of the descriptor ring (TDLEN).
4. If needed, set TDWBAL/TWDBAH to enable head write back.
5. Program the TXDCTL register with the desired TX descriptor write-back policy (see the recommended values in the register description).
6. Enable the queue using the TXDCTL.ENABLE bit. Poll the TXDCTL register until the *Enable* bit is set.

Note: Queue 0 is enabled by default when the DMATXCTL.TE bit is set. If transmit is already enabled (DMATXCTL.TE bit is set), this queue should be disabled (clear TXDCTL.ENABLE bit) before changing the basic queue parameters (TDBAL, TDBAH, TDLEN, TDWBAL/TWDBAH).

Note: The tail register of the queue (TDT) should not be bumped until the queue is enabled.

4.6.8.1.1 Dynamic Enabling and Disabling of Transmit Queues

Transmit queues can be enabled or disabled dynamically given the following procedure is followed.

Enabling:

Follow the per-queue initialization described in the previous section.

Disabling:

1. Stop storing packets for transmission in this queue.
2. The completion of the last transmit descriptor must be visible to software to guarantee that packets are not lost in step 5 ([Section 4.6.8](#)). Therefore, its *RS* bit must be set or *WTHRESH* must be greater than zero. If none of the previous conditions are met, software should add a null Tx data descriptor with an active *RS* bit.
3. Wait until the software head of the queue (*TDH*) equals the software tail (*TDT*) indicating the queue is empty.
4. Wait until all descriptors are written back (polling the *DD* bit in ring or polling the *Head_WB* content). It might be required to flush the transmit queue by setting the *TXDCTL[n].SWFLSH* bit if the *RS* bit in the last fetched descriptor is not set or if *WTHRESH* is greater than zero.
5. Disable the queue by clearing *TXDCTL.ENABLE*.
6. Any packets waiting for transmission in the packet buffer would still be sent at a later time.

The transmit path can be disabled only after all transmit queues are disabled.

4.6.9 Virtualization Initialization Flow

4.6.9.1 VMDq Mode

4.6.9.1.1 Global Filtering and Offload Capabilities

- Select one of the VMDQ pooling and queuing methods
 - Pool Selection can be based on MAC/VLAN, and optionally E-tag. The filtering mode is defined using the *PFVTCTL.POOLING_MODE* field.
- Queue in poll can be based either on DCB or RSS. Potential values for *MRQC.MRQE* to set VMDq modes are 1000b to 1111b.
- DCB should be initiated as described in [Section 4.6.10](#). In RSS mode, the RSS key (*VFRSSRK*) and redirection table (*VFRETA*) of the pool should be programmed. The usage of DCB can be controlled per pool.
- Configure the *PFVTCTL* register to define the default pool.
- Enable replication via the *PFVTCTL.RPL_EN* bit.
- If needed, enable padding of small packets via the *HLREG0.TXPADEN* bit.
- The *MPSAR* registers are used to associate Ethernet MAC Addresses to pools. Using the *MPSAR* registers, software must reprogram *RAL[0]* and *RAH[0]* by their values (software could read these registers and then write them back with the same content).

4.6.9.1.2 Mirroring Rules

For each mirroring rule to be activated:

- Set the type of traffic to be mirrored in the `PFMRCTL[n]` register. Only one type of traffic can be selected in each rule.
- Set the mirror pool by setting the `PFMRCTL[n].MP` bit.
- For pool mirroring, set the `PFMRVM[n]` register with the pools to be mirrored.
- For VLAN mirroring, set the `PFMRVLAN[n]` register with the indexes from the `PFVLVF` registers of the VLANs to be mirrored.

4.6.9.1.3 Security Features

For each pool, the driver might activate the MAC, VLAN and EtherType anti-spoof features via the relevant bit in the `PFVFSPPOOF.MACAS`, `PFVFSPPOOF.VLANAS`, `PFVFSPPOOF.ETHERTYPEAS`, and `PFVFSPPOOF.ETHERTYPELB` fields, respectively.

In addition, stripping and hiding of VLAN and external tag (E-tag) may also be requested via the `PFQDE.HIDE_VLAN` and `PFQDE.STRIP_TAG` flags, respectively.

4.6.9.1.4 Per-Pool Settings

As soon as a pool of queues is associated to a VM, software should set the following parameters:

- Associate the unicast Ethernet MAC Address of the VM by enabling the pool in the `MPSAR` registers.
- If all the Ethernet MAC Addresses are used, the Unicast Hash Table (`PFUTA`) can be used. Pools servicing VMs whose address is in the hash table should be declared as so by setting the `PFVML2FLT.ROPE` bit. Packets received according to this method didn't pass perfect filtering and are indicated as such.
- Enable the pool in all `RAH/RAL` registers representing the multicast Ethernet MAC Addresses this VM belongs to.
- If all the Ethernet MAC Addresses are used, the Multicast Hash Table (`MTA`) can be used. Pools servicing VMs using multicast addresses in the hash table should be declared as so by setting the `PFVML2FLT.ROMPE` bit. Packets received according to this method did not pass perfect filtering and are indicated as such.
- Define whether this VM should get all multicast/broadcast packets in the same VLAN via `PFVML2FLT.MPE` and `PFVML2FLT.BAM`, and whether it should accept un-tagged packets via `PFVML2FLT.AUPE`.
- Enable the pool in each of the `PFVLVF` and `PFVLVFB` registers this VM belongs to.
- A VM might be set to receive it's own traffic in case the source and the destination are in the same pool via the `PFVMTXSW.LLE` bit.
- Whether VLAN header and CRC should be stripped from the packet. Note that even if the CRC is kept, it might not match the actual content of the forwarded packet, because of other offloading applications such as VLAN strip.
 - A striped VLAN may be also hidden from the VM.
- Set which header split is required via the `PSRTYPE` register.

- In RSS mode, define if the pool uses RSS via the proper MRQC.MRQE mode.
 - Enable the Pool in the PFVFRE register to allow Rx Filtering
 - To Enable Multiple Tx queues, Set the MTQC as described in [Section 7.2.1.2.1](#)
 - Enable the Pool in the PFVFTE register to allow Tx Filtering
- Enable Rx and Tx queues as described in [Section 4.6.7](#) and [Section 4.6.8](#).
- For each Rx queue a drop/no drop flag can be set in SRRCTL.DROP_EN and via the PFQDE register, controlling the behavior in cases no receive buffers are available in the queue to receive packets. The usual behavior is to allow a drop to avoid head of line blocking. Setting the PFQDE (per queue) is done by using the QUEUE_INDEX field in the PFQDE register.

4.6.9.2 IOV Initialization

4.6.9.2.1 PF Driver Initialization

The PF driver is responsible for the link setup and handling of all the filtering and offload capabilities for all the VFs as described in [Section 4.6.9.1.1](#) and the security features as described in [Section 4.6.9.1.3](#). It should also set the bandwidth allocation per transmit queue for each VF as described in [Section 4.6.9](#).

Note: Link setup might include the authentication process (802.1X or other) and setup of the DCB parameters.

After all the common parameters are set, the PF driver should set all the VFMailbox[n].RSTD bits by setting the CTRL_EXT.PFRSTD bit.

PF enables VF traffic via the PFVFTE and PFVFRE registers after all VF parameters are set as defined in [Section 4.6.9.1.4](#).

Note: If the operating system changes the NumVF setting in the PCIe SR-IOV Num VFs register after the device was active, it is required to initiate a PF software reset following this change.

4.6.9.2.1.1 VF Specific Reset Coordination

After the PF driver receives an indication of a VF FLR via the PFVFLREC register, it should enable the receive and transmit for the VF only once the device is programmed with the right parameters as defined in [Section 4.6.9.1.4](#). The receive filtering is enabled using the PFVFRE register and the transmit filtering is enabled via the PFVFTE register.

Note: The filtering and offloads setup might be based on a central IT settings or on requests from the VF drivers.

4.6.9.2.2 VF Driver Initialization

At initialization, after the PF indicated that the global initialization was done via the VFMailbox.RSTD bit, the VF driver should communicate with the PF, either via the mailbox or via other software mechanisms to assure that the right parameters of the VF are programmed as described in [Section 4.6.9.1.4](#). The PF driver might then send an acknowledge message with the actual setup done according to the VF request and the IT policy.

The VF driver should then setup the interrupts and the queues as described in [Section 4.6.6](#), [Section 4.6.7](#) and [Section 4.6.8](#).

4.6.9.2.3 Full Reset Coordination

A mechanism is provided to synchronize reset procedures between the PF and the VFs. It is provided specifically for PF software reset but can be used in other reset cases as described later in this section.

The procedure is as follows:

One of the following reset cases takes place:

- LAN Power Good
- PCIe reset (PE_RST_N and in-band)
- D3hot --> D0
- FLR
- Software reset by the PF

The E610 sets the *RSTI* bits in all VFMailbox registers. Once the reset completes, each VF might read its VFMailbox register to identify a reset in progress.

Once the PF completes configuring the device, it clears the CTRL_EXT.PFRSTD bit. As a result, the E610 clears the *RSTI* bits in all VFMailbox registers and sets the *Reset Done (RSTD)* bits are set in all VFMailbox registers.

Until a RSTD condition is detected, the VFs should only access the VFMailbox register and should not attempt to activate the interrupt mechanism or the transmit and receive process.

4.6.10 DCB Configuration

After power-up or device reset, DCB and any type of FC are disabled by default, and a unique TC and packet buffer (such as PB0) is used. In this mode, the host might exchange information via DCX protocol to determine the number of TCs to be configured. Before setting the device to multiple TCs, it should be reset (software reset).

The registers concerned with setting the number of TCs are: RXPBSIZE[0-7], TXPBSIZE[0-7], TXPBTHRESH, MRQC, MTQC, and RTRUP2TC registers as well as the following bits RTRPCS.RAC, RTTDCS.TDPAC, RTTDCS.VMPAC and RTTPCS.TPPAC.

They cannot be modified on the fly, but only after device reset. Packet buffers with a non-null size must be allocated from PB0 and up.

Rate parameters and bandwidth allocation to VMs can be modified on the fly without disturbing traffic flows.

4.6.10.1 CPU Latency Considerations

When the CPU detects an idle period of some length, it enters a low-power sleep state. When traffic arrives from the network, it takes time for the CPU to wake and respond (such as to snoop). During that period, Rx packets are not posted to system memory.

If the entry time-to-sleep state is too short, the CPU might be getting in and out of its sleep state in between packets, therefore impacting latency and throughput. 100 µs is defined as a safe margin for entry time to avoid such effects.

Each time the CPU is in low power, received packets need to be stored (or dropped) in the E610 for the duration of the exit time. Given 64 KB Rx packet buffers per TC in the E610, (i.e. assuming typically 4 TCs per port) PFC does not spread (or the packet is not dropped) provided the CPU exits its low power state within 50 μ s.

4.6.10.2 Link Speed Change Procedure

Each time the link status or speed is changed, hardware automatically updates the rates that were loaded by software relatively to the new link speed. This means that if a rate limiter was set by software to 500 Mb/s for a 10 GbE link speed, it is changed by hardware to 50 Mb/s if the link speed has changed to 1 GbE.

Since rates must be considered as absolute rate limitations (expressed in Mb/s, regardless of the link speed), in such cases software is responsible to either clear all the rate-limiters and/or re-load each rate with the correct value relatively to the new link speed. In the previous example, the new rate value to be loaded by software must be multiplied by 10 to maintain the rate limitation to 500 Mb/s.

4.6.10.3 Initial Configuration Flow

Only the following configuration modes are allowed.

4.6.10.3.1 General Case: DCB-On, VT-On

1. Configure packet buffers, queues, and traffic mapping:

- 8 TCs mode — Packet buffer size and threshold, typically $RXPBSIZE[0-7].SIZE = 0x30, 0x28$, or $0x20$, depending on the size of the flow director table. Refer to rules in [Section 3.6.5.3.5](#) for a non-symmetrical sizing.

$TXPBSIZE[0-7].SIZE = 0x14$

but non-symmetrical sizing is also allowed (see rules in the TXPBSIZE register)

$TXPBTHRESH.THRESH[0-7] = TXPBSIZE[0-7].SIZE$ — Max expected Tx packet length in this TC.

- 4 TCs mode — Packet buffer size and threshold, typically $RXPBSIZE[0-3].SIZE = 0x60, 0x50$, or $0x40$, depending on the size of the flow director table. Refer to rules in [Section 3.6.5.3.5](#) for a non-symmetrical sizing.

$RXPBSIZE[[4-7].SIZE = 0x0$

$TXPBSIZE[0-3].SIZE = 0x28$

$TXPBSIZE[4-7].SIZE = 0x0$

but non-symmetrical sizing among TCs[0-3] is also allowed (see rules in TXPBSIZE register)

$TXPBTHRESH.THRESH[0-3] = TXPBSIZE[0-3].SIZE$ — Maximum expected Tx packet length in this TC.

$TXPBTHRESH.THRESH[4-7] = 0x0$

- Multiple Receive and Transmit Queue Control (MRQC and MTQC)
 - Set $MRQC.MRQE$ to 1xxxb, with the 3 LS bits set according to the number of VFs, TCs, and RSS mode as listed in the MRQC register section.
 - Set both DCB_ENA and VT_ENA bits in the MTQC register.

- Set `MTQC.NUM_TC_OR_Q` according to the number of TCs/VFs enabled as described in the MTQC register section.
 - Set the `PFVTCTL.VT_ENA` bit (as the `MRQC.VT_ENA` bit) Queue Drop Enable (PFQDE): In SR-IOV the `PFQDE` bit should be set to 1b in the `PFQDE` register for all queues. In VMDq mode, the `PFQDE` bit should be set to 0b for all queues.
 - Split Receive Control (`SRRCTL[0-127]`): `DROP_EN` = 1b — drop policy for all the queues, to avoid crosstalk between VMs
 - Rx User Priority (UP)-to-TC (`RTRUP2TC`)
 - Tx UP-to-TC (`RTTUP2TC`)
 - DMA Tx TCP Max Allow Size Requests (`DTXMXSZRQ`) — set `MAX_BYTE_NUM_REQ` = 0x010 = 4 KB.
 - Operate the SEC-TX buffer in DCB mode by setting `SECTXMINIFG.MRKRINSERT` to 0x640 or to 0x1A0 depending if 9.5 KB Jumbo frames are supported or not, respectively.
2. Enable PFC and disable legacy flow control:
- Enables transmit priority flow control via: `FCCFG.TFCE` = 10b, and for the TC(s) where no drop behavior is required by `FCRTH.FCEN` = 1b and appropriate setting in corresponding `FCRTL.RTL` and `FCRTH.RTH`.
 - Note:** `FCRTL.RTL` must be set to a non zero value when `FCRTH.FCEN` = 1b.
 - Enables receive priority flow control via `MFLCN.RPFCM` = 1b and `MFLCN.RPFCE` set for the UP(s) where no drop behavior is required.
 - Disable receive legacy flow control via `MFLCN.RFCE` = 0b.
 - Refer to [Section 8.2.2.3](#) for other registers concerned with flow control
3. Configure arbiters, per TC[0-1]:
- Tx Descriptor Plane T1 Config (`RTTDT1C`) per queue, via setting `RTTDQSEL` first
 - Tx Descriptor Plane T2 Config (`RTTDT2C[0-7]`)
 - Tx Packet Plane T2 Config (`RTTPT2C[0-7]`)
 - Rx Packet Plane T4 Config (`RTRPT4C[0-7]`)
4. Enable TC and VM arbitration layers:
- Tx Descriptor Plane Control and Status (`RTTDCS`), bits: `TDPAC` = 1b, `VMPAC` = 1, `TDRM` = 1b, `BDPM` = 0b, `BPBFSM` = 0b.
 - Tx Packet Plane Control and Status (`RTTPCS`): `TPPAC` = 1b, `TPRM` = 1b, `ARBD` = 0x004, `BYPASS_DCB_ARB` = 0
 - Rx Packet Plane Control and Status (`RTRPCS`): `RAC` = 1b, `RRM` = 1b

4.6.10.3.2 DCB-On, VT-Off

Set the configuration bits as specified in [Section 4.6.10.3.1](#) with the following exceptions:

- Configure packet buffers, queues, and traffic mapping:
 - MRQC and MTQC
 - Set *MRQE* to 0xxxb, with the 3 LS bits set according to the number of TCs, and RSS mode
 - Set the *DCB_ENA* bit and clear the *VT_ENA* bit in the MTQC register.
 - Set *MTQC.NUM_TC_OR_Q* according to the number of TCs enabled
 - Clear the *PFVTCTL.VT_ENA* bit (as the *MRQC.VT_ENA* bit)
- Allow no-drop policy in Rx:
 - *PFQDE*: The *PFQDE* bit should be set to 0b in the *PFQDE* register for all queues enabling per queue policy by the *SRRCTL[n]* setting.
 - Split Receive Control (*SRRCTL[0-127]*): The *DROP_EN* bit should be set (per receive queue) according to the required drop / no-drop policy of the TC of the queue.
- Disable VM arbitration layer:
 - Clear the *RTTDT1C* register, per each queue, via setting *RTTDQSEL* first
 - *RTTDCS.VMPAC* = 0b

4.6.10.3.3 DCB-Off, VT-On

Set the configuration bits as specified in [Section 4.6.10.3.1](#) with the following exceptions:

- Disable multiple packet buffers and allocate all queues to PB0:
 - *RXPBSIZE[0].SIZE* = 0x180, *RXPBSIZE[1-7].SIZE* = 0x0
 - *TXPBSIZE[0].SIZE* = 0xA0, *TXPBSIZE[1-7].SIZE* = 0x0
 - *TXPBTHRESH.THRESH[0]* = 0xA0 — Max expected Tx packet length in this TC
TXPBTHRESH.THRESH[1-7] = 0x0
 - MRQC and MTQC
 - Set *MRQE* to 1xxxb, with the 3 LS bits set according to the number of VFs and RSS mode
 - Clear *DCB_ENA* bit and set the *VT_ENA* bit in the MTQC register.
 - Set *MTQC.NUM_TC_OR_Q* according to the number of VFs enabled
 - Set the *PFVTCTL.VT_ENA* bit (as the *MRQC.VT_ENA* bit) Rx UP-to-TC (*RTRUP2TC*), *UPnMAP*=0, *n*=0,...,7
 - Tx UP-to-TC (*RTTUP2TC*), *UPnMAP*=0, *n*=0,...,7
 - DMA Tx TCP Max Allow Size Requests (*DTXMXSZRQ*) — set *MAX_BYTE_NUM_REQ* = 0xFFFF = 1 MB
 - Operate the SEC-TX buffer in non-DCB mode by setting *SECTXMINIFG.MRKRINSERT* to 0x10
- Disable PFC and enabled legacy flow control:
 - Disable receive priority flow control via: *MFLCN.RPFCM* = 0b and *MFLCN.RPFCE[7:0]* = 0x00
 - Enable transmit legacy flow control via: *FCCFG.TFCE* = 01b

- Enable receive legacy flow control via: $MFLCN.RFCE = 1b$
- Configure VM arbiters only, reset others:
 - Tx Descriptor Plane T1 Config (RTTDT1C) per pool, via setting RTTDQSEL first for the pool index. Clear RTTDT1C for other queues.
 - Clear RTTDT2C[0-7] registers
 - Clear RTTPT2C[0-7] registers
 - Clear RTRPT4C[0-7] registers
- Disable TC arbitrations while enabling the PB Free Space Monitor:
 - Tx Descriptor Plane Control and Status (RTTDCS), bits: $TDPAC = 0b$, $VMPAC = 1b$, $TDRM = 0b$, $BDPM = 1b$, $BPBFSM = 0b$
 - Tx Packet Plane Control and Status (RTTPCS): $TPPAC = 0b$, $TPRM = 0b$, $ARBD = 0x24$, $BYPASS\_DCB\_ARB = 1$.
 - Rx Packet Plane Control and Status (RTRPCS): $RAC = 0b$, $RRM = 0b$

4.6.10.3.4 DCB-Off, VT-Off

Set the configuration bits as specified in [Section 4.6.10.3.1](#) with the following exceptions:

- Disable multiple packet buffers and allocate all queues and traffic to PB0:
 - $RXPBSIZE[0].SIZE = 0x180$, $RXPBSIZE[1-7].SIZE = 0x0$
 - $TXPBSIZE[0].SIZE = 0xA0$, $TXPBSIZE[1-7].SIZE = 0x0$
 - $TXPBTHRESH.THRESH[0] = 0xA0$ — Max expected Tx packet length in this TC
 $TXPBTHRESH.THRESH[1-7] = 0x0$
 - MRQC and MTQC
 - Set $MRQE$ to $0xxx$, with the 3 LS bits set according to the RSS mode
 - Clear both DCB_ENA and VT_ENA bits in the MTQC register.
 - Set $MTQC.NUM\_TC\_OR\_Q$ to $00b$.
 - Clear the $PFVTCTL.VT\_ENA$ bit (as the $MRQC.VT\_ENA$ bit) Rx UP-to-TC (RTRUP2TC), $UPnMAP=0$, $n=0,...,7$
 - Tx UP-to-TC (RTTUP2TC), $UPnMAP=0$, $n=0,...,7$
 - DMA Tx TCP Max Allow Size Requests (DTXMXSZRQ) — set $MAX\_BYTE\_NUM\_REQ = 0xFFFF = 1\text{ MB}$
 - Operate the SEC-TX buffer in non-DCB mode by setting $SECTXMINIFG.MRKRINSERT$ to $0x10$
- Allow no-drop policy in Rx:
 - PFQDE: The $PFQDE$ bit should be set to $0b$ in the PFQDE register for all queues enabling per queue policy by the $SRRCTL[n]$ setting.
 - Split Receive Control ($SRRCTL[0-127]$): The $DROP\_EN$ bit should be set (per receive queue) according to the required drop / no-drop policy of the TC of the queue.

- Disable PFC and enable legacy flow control:
 - Disable receive priority flow control via: `MFLCN.RPFCM` = 0b and `MFLCN.RPFCE[7:0]` = 0x00
 - Enable receive legacy flow control via: `MFLCN.RFCE` = 1b
 - Enable transmit legacy flow control via: `FCCFG.TFCE` = 01b
- Reset all arbiters:
 - Clear the RTTDT1C register, per each queue, via setting RTTDQSEL first
 - Clear RTTDT2C[0-7] registers
 - Clear RTTPT2C[0-7] registers
 - Clear RTRPT4C[0-7] registers
- Disable TC and VM arbitration layers:
 - Tx Descriptor Plane Control and Status (RTTDCS), bits: `TDPAC` = 0b, `VMPAC` = 0b, `TDRM` = 0b, `BDPM` = 1b, `BPBFSM` = 1b
 - Tx Packet Plane Control and Status (RTTPCS): `TPPAC` = 0b, `TPRM` = 0b, `ARBD` = 0x224
 - Rx Packet Plane Control and Status (RTRPCS): `RAC` = 0b, `RRM` = 0b

4.6.10.4 Configuration Rules

4.6.10.4.1 TC Parameters

Traffic Class (TC):

Per 802.1p priority #7 is the highest priority.

A specific TC can be configured to receive or transmit a specific amount of the total bandwidth available per port.

Bandwidth allocation is defined as a fraction of the total available bandwidth, which can be less than the full Ethernet link bandwidth (if it is bounded by the PCIe bandwidth or by flow control).

Low latency TC should be configured to use the highest priority TC possible (TC 6, 7); the lowest latency is achieved using TC7.

Bandwidth Groups (BWGs):

BWGs are used to represent different traffic types. A traffic type (such as storage, IPC LAN, or manageability) can have more than one TC. For example, one for control traffic and one for the raw data. By grouping these two TCs to a BWG, the end user can allocate bandwidth to the storage traffic so that unused bandwidth by the control could be used by the data and vice versa. This BWG concept supports the converged fabric as each traffic type, that's used to run on a different fabric, can be configured as a BWG and gets its resources as if it was on a different fabric.

1. To configure DCB not to share bandwidth between TCs, each TC should be configured as a separate BWG.
2. There are no limits on the TCs that can be bundled together as a BWG. All TCs can be configured as a single BWG.
3. BWG numbers should be sequential starting from zero until the total number of BWG minus one.
4. BWG numbers do not imply priority; priority is only set according to TCs.

Refill Credits:

Refill credits regulate the bandwidth allocated to BWGs and TCs. The ratio between the credits of the BWG's represents the relative bandwidth percentage allocated to each BWG. The ratio between the credits of the TCs represents the relative bandwidth percentage allocated to each TC within a BWG.

Credits are configured and calculated using 64-byte granularity.

1. In any case, the number of refill credits assigned per TC should be as small as possible but still larger than the maximum frame size used and larger than 1.5 KB. Using a lower refill value causes more refill cycles before a packet can be sent. These extra cycles unnecessarily increase the latency.
2. Refill credits ratio between TCs should be equal to the desired ratio of bandwidth allocation between the different TCs. Applying rule #1, means bandwidth shares are sorted from the smaller to the bigger, and just one maximum sized frame is allocated to the smallest.
3. The ratio between the refill credits of any two TCs should not be greater than 100.
4. Exception to rule #2 — TCs that require low latency should be configured so that they are under subscribed. For example, the credit refill value should provide these TCs somewhat more bandwidth than what they actually need. Low latency TCs should always have credits so they can be next in line for WSP arbitration.

This exception causes the low latency TC to always have maximum credits (as it starts with maximum credits and on average cycle uses less than the refill credits).

The end point that is sending/receiving packets of 127 bytes eventually get double the bandwidth it was configured to, as all credits calculated by rounding the values down to the next 64 byte-aligned value.

Max Credit Limit:

The maximum credit limit value establishes a limit for the number of credits that a TC or BWG can own at any given time. This value prevents stacking up stale credits that can be added up over a relatively long period of time and then used by TCs all at once, altering fairness and latency.

Max credit limits are configured and calculated using 64-byte granularity.

1. Maximum credit limit should be bigger than the refill credits allocated to the TC.
2. Maximum limit should be set to be as low as possible while still meeting other rules to minimize the latency impact on low latency TCs.
3. If a low latency TC generates a burst that is larger than its maximum credit limit, this TC might experience higher latency since the TC needs to wait for allocation of additional credits because it finished all its credits for this cycle. Therefore, maximum credit limit for a low latency TC must be set bigger than the maximum burst length of traffic expected on that TC — for all VMs at once. If TC7 and TC6 are for low latency traffic, it leads to:

$$\text{Max}(\text{TC7},6) \geq \text{MaxBurst}(\text{TC7},6) \text{ served with low latency}$$

4. An arbitration cycle can be extended when one or more TCs accumulate credits more than their refill values (up to their maximum credit limit). For such a case a low latency TC should be provided with enough credits to cover for the extended cycle duration. Since the low latency TC operates at maximum credits (see rule #3) its maximum credit limit should meet the following formula.

$$\{\text{Max}(\text{TCx})/\text{SUMi}=0..7[\text{Max}(\text{TCi})]\} \geq \{\text{BW}(\text{TCx})/\text{Full BW}\}$$

5. The formula applies to both the descriptor arbiter and data arbiter.

When in a virtualized environment, the “low latency TC condition” checked by the VM WRR arbiter (see [Section 7.6.2.3.2](#)) induces the following relation between the max credits of a low latency TC and the refill credits of its attached VM arbiter:

$$\text{Max}(\text{TCx}) \geq 2 \times \{\text{SUMi}=0\ldots 15[\text{Refill}(\text{VMi})]\}$$

6. To ensure bandwidth for a low priority TC (when those are allocated with most of the bandwidth) the maximum credit value of the low priority TC in the data arbiter needs to be high enough to ensure sync between the two arbiters. In the following equation the bandwidth numbers are from the descriptor arbiter while the maximum values are of the data arbiter.

$$\{\text{Max}(\text{TCx})/\text{SUMi}=x+1\ldots 7[\text{Max}(\text{TCi})]\} \geq \{\text{BW}(\text{TCx})/\text{Full\_PCIE\_BW}\}$$

Note that the previous equation is worst case and covers the assumption that all higher TCs have the full maximum to transmit.

Tip: A simplified maximum credits allocation scheme would be to find the minimum number $N \geq 2$ such that rules #3 and #5 are respected, and allocate:

$$\text{Max}(\text{TCi}) = N \times \text{Refill}(\text{TCi}), \text{ for } i=0\ldots 7$$

By maintaining the same ratios between the maximum credits and the bandwidth shares, the bandwidth allocation scheme is made more immune to disturbing events such as receiving priority pause frames with short timer values.

GSP and LSP:

Bit TC.LSP (TC Link Strict Priority): This bit specifies that the configured TC can transmit without any restriction of credits. This effectively means that the TC can take up the entire link bandwidth, unless preempted by higher priority traffic. The Tx queues associated with a LSP TC must be set as strict low latency in the TXLLQ[n] registers.

Bit TC.GSP (TC Strict Priority within Group): This bit defines whether strict priority is enabled or disabled for this TC within its BWG. If bit TC.GSP is set to 1b, the TC is scheduled for transmission using strict priority. It does not check for availability of credits in the TC. It does; however, check whether the BWG of this TC has credits. For example, the amount of traffic generated from this TC is still limited by the bandwidth allocated for the BWG.

1. TCs with the *LSP* bit set should be the first to be considered by the scheduler. This implies that the *LSP* bit should be configured to the highest priority TCs. For example, starting from priority 7 and down; the other TCs should be used for groups with bandwidth allocation. It is recommended to use *LSP* only for one TC (TC7) as the first LSP TC takes its bandwidth and there are no guarantees to the lower priority LSPs.
2. The *GSP* bit can be set to more than one TC in a BWG, always from the highest priority TC within that BWG downward. For the LAN scenario, all TCs could be configured to be GSP as their bandwidth needs are not known.
3. For a low latency TC where the *GSP* bit is set, non-null refill credits must be set for at least one maximum-sized frame. This ensures that even after its been quiet for a while, some BWG credits are left available to the GSP TC, for serving it with minimum latency (without waiting for replenishing). Bigger refill credits values ensure longer bursts of GSP traffic served with minimum latency.

4.6.10.4.2 VM Parameters

Refill Credits:

Refill credits regulate the fraction of the TC's bandwidth that is allocated to a VM. The ratio between the credits of the VMs represent the relative TC bandwidth percentage allocated to each VM.

Credits are configured and calculated using 64-byte granularity.

1. In any case, the number of refill credits assigned per VM should be as small as possible but still larger than the maximum frame size used and larger than 1.5 KB. Using a lower refill value causes more refill cycles before a packet can be sent. These extra cycles increase the latency unnecessarily.
2. The refill credits ratio between VMs should be equal to the desired ratio of bandwidth allocation between the different TCs. Applying rule #1, means bandwidth shares are sorted from the smaller to the bigger, and just one maximum sized frame is allocated to the smallest.
3. The ratio between the refill credits of any two VMs within the TC should not be greater than 10.

VMs that send/receive packets of 127 bytes eventually get double the bandwidth it was configured to as all credits are calculated by rounding the values down to the next 64 byte-aligned value.

Refill and Maximum Credits (MaxCredits) Setting Example:

This example assumes a system with only four TCs and three VMs present, along with the following bandwidth allocation scheme. Also, note that full PCIe bandwidth is evaluated to 15 G.

Table 4-15. Bandwidth Share Example

TCs and VMs		Bandwidth Share %	Notes
TC0	Total	40	9.5 KB jumbo frames allowed.
	VM0	60	
	VM1	30	
	VM2	10	
TC1	Total	20	No jumbo frames.
	VM0	34	
	VM1	33	
	VM2	33	
TC2	Total	30	Low latency TC. No jumbo frames. Bandwidth share already increased. MaxBurstTC2=120 KB
	VM0	80	
	VM1	10	
	VM2	10	
TC3	Total	10	Low latency LSP TC. No jumbo frames. MaxBurstTC3=36 KB
	VM0	20	
	VM1	60	
	VM2	20	

The ratios between TC refills are driven by TC0, which was set as 152 for supporting 9.5 KB jumbo frames.

The ratio between MaxCredits and Refill were taken as 17 for all the TCs, as driven by the TC2 relation between MaxCredits and MaxBurstTC2.

Table 4-16. Refill and MaxCredits Settings

TCs and VMs		Refill (64-Byte Units)	MaxCredits (64-Byte Units)
TC0	Total	152	2584
	VM0	912	
	VM1	456	
	VM2	152	
TC1	Total	76	1292
	VM0	25	
	VM1	24	
	VM2	24	
TC2	Total	114	1938
	VM0	192	
	VM1	24	
	VM2	24	
TC3	Total	38	646
	VM0	24	
	VM1	72	
	VM2	24	

4.6.10.4.3 Rate-Limiters

It might be useful to setup rate limiters on Tx queues (rate-limiting VF traffic). Setting a rate-limiter on Tx queue N to a Target Rate requires the following settings:

- Use of advanced Tx descriptors
- `RTTQCNRM.MMW_SIZE` set to the amount of back-to-back compensation to be granted to a quiet item.
- `RTTDQSEL.TXQ_IDX` set to N.
- Compute the Rate Factor = Link Speed / Target Rate, where Link Speed is either 10 Gb/s or 1 Gb/s. The fractional binary number obtained is broken down into its 2^n components as follows:

$$b_9b_8b_7b_6b_5b_4b_3b_2b_1b_0.b_{-1}b_{-2}b_{-3}b_{-4}b_{-5}b_{-6}b_{-7}b_{-8}b_{-9}b_{-10}b_{-11}b_{-12}b_{-13}b_{-14}$$

where b_n is the binary coefficient of the 2^n component.

- Set RTTQCNRC register with
 - Set `RTTQCNRC[13:0] = RTTQCNRC.RF_DEC[13:0] = [b_{-1}b_{-2}b_{-3}b_{-4}b_{-5}b_{-6}b_{-7}b_{-8}b_{-9}b_{-10}b_{-11}b_{-12}b_{-13}b_{-14}]`
 - `RTTQCNRC[23:14] = RTTQCNRC.RF_INT[9:0] = [b_9b_8b_7b_6b_5b_4b_3b_2b_1b_0]`
 - `RTTQCNRC[31] = RTTQCNRC.RS_ENA = 1b`

Numerical Example:

- Target Rate = 4 Gb/s
 - Link Speed = 10 Gb/s
- Rate Factor = $10 / 4 = 2.5 = 1 \times 2^1 + 1 \times 2^{-1} = 0000000010.10000000000000b$
- RTTQCNRC[13:0] = [10000000000000]
 - RTTQCNRC[23:14] = [0000000010]
 - RTTQCNRC = [1000 0000 0000 0000 1010 0000 0000 0000] = 0x8000A000

4.7 Access to Shared Resources

Part of the resources in the E610 are shared between several software entities — namely the software device drivers of the two ports and internal firmware. In order to avoid contentions, a software device driver that needs to access one of these resources should use the Gaining Control of Shared Resource by Software flow described in [Section 10.11.3.4.1](#) in order to acquire ownership of this resource, and use the Releasing a Shared Resource by Software flow described there in order to relinquish ownership of this resource.

The shared resources are:

- NVM
- MAC CSRs.
- I²C interface of each port
- MDIO interface of each port

4.8 BIOS Init

This section describes how BIOS code might update the E610's configuration and the capabilities for network boot. The following sections are provided:

- [Section 4.8.2](#) describes how BIOS code might change the E610's configuration on each boot
- [Section 4.8.3](#) describes some aspects of supporting network boot
- [Section 4.8.4](#) describes the specific setting of each of the E610's components required for the BIOS Init stage

4.8.1 Initial State

The state of the E610 when BIOS begins to access it is described in [Section 4.8.4](#).

BIOS code must check that the EMP completed device initialization. This is done through the Get Version AQ command described in [Section 10.11.2.7.1](#).

4.8.2 Non-Persistent Configuration

During power up, the E610 is factory-configured from the NVM. However, the factory configuration might be overridden in two possible ways:

- **Persistent configuration** — System tools (such as software agents and SMCLP commands) might write into the NVM and change the factory defaults. It is also possible to add alternate values into the NVM so that the factory defaults are preserved and might be restored at a later time.
- **Non-persistent configuration** — An on-die Alternate RAM structure is provided to store configuration values which are not supposed to be maintained between cold resets. The information in this structure is retained during all resets other than a cold reset of the device. The specific fields are loaded by the device firmware according to [Table 4-17](#).

During a cold reset sequence, the alternate structure might be written by either an external system management agent (via the device management interfaces) or by BIOS level code (like SMASH/CLP commands).

4.8.2.1 Alternate RAM Structure

The alternate structure is 6 KB, partitioned into 32-bit entries. Accessing the structure is done by addressing 32-bit entries. An address is therefore 11 bits (1.5 K DWords), where address 0x000 points to the beginning of the memory.

The structure is partitioned as follows:

Section	Address	Size (DW)	Content
Per PF	0 – 128	128 - 64 per PF	8 per-PF sections, one per PF. These sections are described in Section 4.8.2.1.1 and can be accessed by each PF.
Reserved	129 - 1023		Reserved for more functions.
Per PF	1024 - 1536	612	Reserved for EMP use, including error logging. Can be written and read only by the EMP. In debug mode, this section can be read by the PFs.

4.8.2.1.1 Per PF Sections

Table 4-17. Per PF Alt RAM Content

Scope	Address (DW)	Contents	Done Alternate Write Required ¹	NVM Dump Required ²	Load Conditions
PF	0	Current LAN MAC Address (Low)	No	No	PFR, CORER
PF	1	Current LAN MAC Address (High)	No	No	PFR, CORER
PF	2-16	Reserved	N/A	N/A	Never
PF	17	Reserved	N/A	N/A	Never
PF	18	Reserved	N/A	N/A	Never
PF	19	Reserved	N/A	N/A	Never
PF	20-63	Reserved	N/A	N/A	Never

1. The "Done Alternate Write Required" column indicates whether the Done Alternate Write (Opcode: 0x0904) command needs to be issued to the device for the configuration to take place on the conditions specified by the "Load Conditions" column. If "Done Alternate Write" is not required, then just writing the value into the Alternate RAM and meeting the load conditions is sufficient for the configuration to take place. If the "Done Alternate Write" command was required, but not sent, the configuration will not be applied, even if the condition is met. Issuing the "Done Alternate Write" command when it is not explicitly required will not have any adverse effects on the operation of the device.
2. Some changes are persistent. To make sure the data is stored in NVM, the shadow RAM shall be written back to the flash using the NVM write Activate command ([Section 3.3.10.8](#)).

4.8.2.1.1.1 Current LAN MAC Address (Offset 0x0, 0x1)

Lower DWord:

Field	Bit(s)	Description
MAC Address	31:0	MAC Address Contains the LS 32-bits of the address.

Upper DWord:

Field	Bit(s)	Description
MAC Address	15:0	MAC Address Contains the MS 16-bits of the address.
Reserved	30:16	Reserved.
Valid	31	Valid bit 0b = The MAC Address two DWords are invalid and should be skipped. 1b = The MAC Address two DWords are valid and should be processed.

Actions taken on a change in this entry:

- When valid, it overrides the MAC Address loaded from the NVM.

4.8.2.2 AQ Commands

Table 4-18 lists the different AQ CSR based commands used to manage the alternate structure.

Table 4-18. List of AQ commands for the Alternate Structure

Command	Opcode	Description	Section Reference
Write Alternate - Direct	0x0900	Write up to two parameters into the alternate structure.	4.8.2.2.1
Write Alternate - Indirect	0x0901	Write a block of parameters into the alternate structure.	4.8.2.2.2
Read Alternate - Direct	0x0902	Read up to two parameters from the alternate structure.	4.8.2.2.3
Read Alternate - Indirect	0x0903	Read a block of parameters from the alternate structure.	4.8.2.2.4
Done Alternate Write	0x0904	Indication that all CLP strings (for the entire E610 in legacy BIOS mode, and per LAN Port in UEFI mode) have been sent to the E610.	4.8.2.2.5
Clear Port Alternate	0x0906	Clear content of Alternate RAM relevant to this port.	4.8.2.2.6

4.8.2.2.1 Write Alternate - Direct (0x0900)

The Write Alternate - Direct command writes to the alternate structure up to two parameters.

Table 4-19. Write Alternate - Direct Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 10.11.2.4.1.1 for details.
Opcode	2-3	0x0900	Command opcode.
Datalen	4-5	0x0	N/A
Return Value/VFID	6-7	0x0	N/A
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
First Parameter Address	16-19		Accessing the alternate structure is done by addressing 32-bit entries.
First Parameter Data	20-23		
Second Parameter Address	24-27		Address should be within the range allocated to the function inside the alternate structure. Value of 0xFF..FF means only the first parameter is written. Accessing the alternate structure is done by addressing 32-bit entries.
Second Parameter Data	28-31		

The following completion is sent for the Write Alternate - Direct command:

Table 4-20. Completion for the Write Alternate - Direct Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 10.11.2.4.1.2 for details.
Opcode	2-3	0x0900	Command opcode.
Datalen	4-5	0x0	N/A
Return Value/VFID	6-7		Some comments on specific errors: 0x0 = No error. ENOMEM = Out of memory (access outside the alternate structure). EACCES = Permission denied (access to another PF's area).
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16-31		Reserved. Might contain the values sent in the original command.

4.8.2.2.2 Write Alternate - Indirect (0x0901)

The Write Alternate - Indirect command writes a block of parameters to the alternate structure. The command defines the number of DWords to be written and the starting address inside the alternate structure.

Table 4-21. Write Alternate - Indirect Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 10.11.2.4.2.1 for details.
Opcode	2-3	0x0901	Command opcode.
Datalen	4-5		Size of buffer accompanying the command (in bytes).
Return Value/VFID	6-7	0x0	N/A
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
Alternate Structure Address	16-19		Lowest address to be written into the alternate structure. Accessing the alternate structure is done by addressing 32-bit entries.
Alternate Structure Length	20-23		Number of DWords to be written into the alternate structure.
Data Address High	24-27	Buffer Address	High bits of the buffer address.
Data Address Low	28-31	Buffer Address	Low bits of the buffer address.

The following completion is sent for the Write Alternate - Indirect command:

Table 4-22. Completion for the Write Alternate - Indirect Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 10.11.2.4.2.2 for details.
Opcode	2-3	0x0901	Command opcode.
Datalen	4-5	0x0	N/A
Return Value/VFID	6-7		Some comments on specific errors: 0x0 = No error. ENOMEM = Out of memory (access outside the alternate structure). EACCES = Permission denied (access to another PF's area).
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16-31		Reserved.

4.8.2.2.3 Read Alternate - Direct (0x0902)

The Read Alternate - Direct command reads from the alternate structure up to two parameters. This command, and the next one, returns the values as stored in the Alternate RAM, which might not reflect the current hardware configuration.

Table 4-23. Read Alternate - Direct Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 10.11.2.4.1.1 for details.
Opcode	2-3	0x0902	Command opcode.
Datalen	4-5	0x0	N/A
Return Value/VFID	6-7	0x0	N/A
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
First Parameter Address	16-19		Address should be within the range allocated to the function inside the alternate structure. Accessing the alternate structure is done by addressing 32-bit entries.
Reserved	20-23	0x0	Reserved.
Second Parameter Address	24-27		Address should be within the range allocated to the function inside the alternate structure. Value of 0xFF..FF means only the first parameter is read. Accessing the alternate structure is done by addressing 32-bit entries.
Reserved	28-31	0x0	Reserved.

The following completion is sent for the Read Alternate - Direct command:

Table 4-24. Completion for the Read Alternate - Direct Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 10.11.2.4.1.2 for details.
Opcode	2-3	0x0902	Command opcode.
Datalen	4-5	0x0	N/A
Return Value/VFID	6-7		Some comments on specific errors: 0x0 = No error. ENOMEM = Out of memory (access outside the alternate structure). EACCES = Permission denied (access outside of PF's area).
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
First Parameter Address	16-19		Copied from the command.
First Parameter Data	20-23		Data read.
Second Parameter Address	24-27		Copied from command. Value of 0xFF..FF means only the first parameter is read.
Second Parameter Data	28-31		Data read.

4.8.2.2.4 Read Alternate - Indirect (0x0903)

The Read Alternate - Indirect command reads a block of parameters to the alternate structure. The command defines the number of DWords to be read and the starting address inside the alternate structure.

Table 4-25. Read Alternate - Indirect Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 10.11.2.4.2.1 for details.
Opcode	2-3	0x0903	Command opcode.
Datalen	4-5		Size of buffer accompanying the command (in bytes).
Return Value/VFID	6-7	0x0	N/A
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
Alternate Structure Address	16-19		Lowest address to be read from the alternate structure. Accessing the alternate structure is done by addressing 32-bit entries.
Alternate Structure Length	20-23		Number of DWords to be read from the alternate structure.
Data Address High	24-27	Buffer Address	High bits of the buffer address.
Data Address Low	28-31	Buffer Address	Low bits of the buffer address.

The following completion is sent for the Read Alternate - Indirect command:

Table 4-26. Completion for the Read Alternate - Indirect Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 10.11.2.4.2.2 for details.
Opcode	2-3	0x0903	Command opcode.
Datalen	4-5	0x0	Actual length of data returned by the command.
Return Value/VFID	6-7		Some comments on specific errors: 0x0 = No error. ENOMEM = Out of memory (access outside the alternate structure). EACCES = Permission denied (access to another PF's area).
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
Alternate Structure Address	16-19		Lowest address read from the alternate structure.
Alternate Structure Length	20-23		Number of DWords read from the alternate structure.
Data Address High	24-27	Buffer Address	High bits of the buffer address.
Data Address Low	28-31	Buffer Address	Low bits of the buffer address.

4.8.2.2.5 Done Alternate Write (0x0904)

The Done Alternate Write command indicates to the E610 that the CLP strings have been sent to it:

- **Legacy BIOS mode** — Sent once per device after all CLP strings have been sent to the E610 (for all LAN ports). Following the command, firmware loads the contents of the alternate structure into the device functional units.
- **UEFI mode** — Sent once per each enabled LAN port. Once the command is received from all enabled ports, firmware loads the contents of the alternate structure into the E610's functional units.

After this command, the software should trigger a Shadow RAM dump using a dummy NVM update with *LAST* flag set (Bit 19.0 in [Table 3-28](#)).

Table 4-27. Done Alternate Write Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 10.11.2.4.2.1 for details.
Opcode	2-3	0x0904	Command opcode.
Datalen	4-5	0x0	N/A
Return Value/VFID	6-7	0x0	N/A
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
BIOS Mode	16.0	See remarks	0b = Legacy BIOS 1b = UEFI
Reserved	16.1-31	0x0	Reserved.

The following completion is sent for the Done Alternate Write command:

Table 4-28. Completion for the Done Alternate Write Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 10.11.2.4.2.2 for details.
Opcode	2-3	0x0904	Command opcode.
Datalen	4-5	0x0	N/A
Return Value/VFID	6-7		ENOSPC = More than 128 VFs are requested.
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16.0	0b	Reserved.
Return Flags	16.1		Reset Needed When set, indicates that software should do a global reset for the Alternate RAM content to take effect.
Reserved	16.1-31	0x0	Reserved.

4.8.2.2.6 Clear Port Alternate Write (0x0906)

The Clear Port Alternate command indicates to the E610 to clear the alternate sections of the PF. The port is inferred from the PF that sent the command.

Table 4-29. Clear Port Alternate Write Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 10.11.2.4.2.1 for details.
Opcode	2-3	0x0906	Command opcode.
Datalen	4-5	0x0	N/A
Return Value/VFID	6-7	0x0	N/A
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16-31	0x0	Reserved.

The following completion is sent for the Clear Port Alternate Write command:

Table 4-30. Completion for the Clear Port Alternate Write Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 10.11.2.4.2.2 for details.
Opcode	2-3	0x0906	Command opcode.
Datalen	4-5	0x0	N/A
Return Value/VFID	6-7	0x0	N/A
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16-31	0x0	Reserved.

4.8.2.3 Processing the Alternate Structure

As a result of each Done Alternate Write AQ command, the EMP loads the relevant alternate structure parameters into the E610 according to the following sequence:

- EMP loads any required parameters from the alternate structure into the hardware.
- EMP ACKs the Done Alternate Write command.
- MAC and MASTER reset is performed.
 - Initiated by software.
 - Reset is done only once and not per each instance of the Done Alternate Write AQ command.
- Hardware is initialized, including loading of the relevant sections of the Alternate Structure into the hardware.

4.8.3 Network Boot

Any functions can be a PXE or UEFI boot function (up to 8 such functions).

4.8.4 Device State

This section is limited to changes in the E610's state made in the BIOS initialization stage. For units not mentioned here, behavior is as described in [Section 4.6.2](#).

4.9 Driver Load

4.9.1 Introduction

4.9.1.1 Driver Load (Non-Virtualized)

As described earlier in this section, by the time the device driver loads, the NVM configuration is already complete and PCIe configuration has taken place. During a device driver load, the following sequence of commands is typically issued to the E610 to initialize it for normal operation. The major initialization steps are:

1. Device driver probes the E610 for resource allocations.
2. Disable interrupts.
3. Initialize the admin queue.
4. Initialize MAC/PHY.
5. Initialize power management (see [Section 5.4](#)).
6. Initialize statistics by reading the counter's initial values to serve as a baseline.
7. Initialize 1588.
8. Enable interrupts.

At this point, the E610 is ready to initialize LAN flows. These are done dynamically during operation:

- Initialize LAN QP, including its interrupts.
- Configure filters (see [Section 7.1.2](#)).

Chapter 5 Power Management and Delivery

This section defines how power management is implemented in the E610.

5.1 Power Targets and Power Delivery

See [Section 11.4](#) for the current consumption and [Section 11.3.1](#) for the power supply specification.

5.2 Power Management

5.2.1 Introduction to E610 Power States

The E610 supports the D0 and D3 power states defined in the PCI Power Management and PCIe specifications. D0 is divided into two sub-states: D0u (D0 un-initialized), and D0a (D0 active). In addition, the E610 supports a Dr state that is entered when PE_RST_N pin is asserted. Dr state is a superset of initial device state and D3 cold.

[Figure 5-1](#) shows the power states and transitions between them.

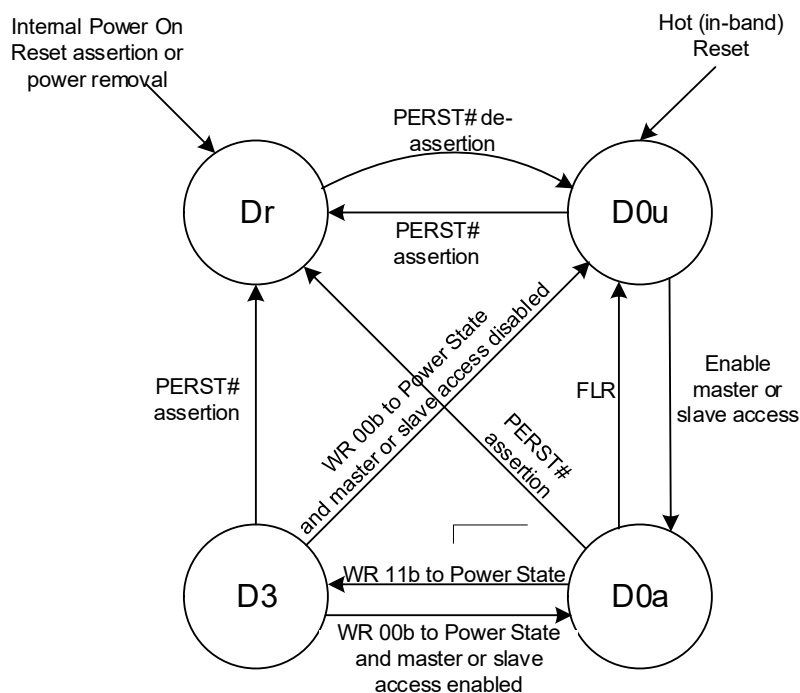


Figure 5-1. Power Management State Diagram

5.2.2 Auxiliary Power Usage

The E610 uses the *AUX_PWR* indication that auxiliary power is available to the controller, and therefore advertises D3cold wake-up support in the *PMC.PME_Support* field and sets the *Aux Power Detected* bit in the PCIe capability structure *Device Status* Register. The *AUX_PWR* pin is strapped during Power On Reset (POR). The amount of power required for the function, which includes the entire Network Interface Card (NIC), is advertised in the Power Management Data register, which is loaded from the NVM.

The only effect of setting *AUX_PWR* to 1b is advertising D3cold wake-up support and changing the reset function of *PME_En* and *PME_Status*. *AUX_PWR* is a strapping option in the E610. If D3cold is supported, the *PME_En* and *PME_Status* bits of the Power Management Control/Status Register (PMCSR), as well as their shadow bits in the Wake-Up Control Register (WUC) are reset only by the power up reset (detection of power rising). The E610 tracks the *PME_En* bit of the PMCSR and the Auxiliary (AUX) power *PM Enable* bit of the PCIe Device Control register to determine the power it can consume (and therefore its power state) in the D3cold state (internal Dr state). Note that the actual amount of power differs between form factors.

According to the following settings the E610 might consume higher auxiliary power than is allowed by PCIe specifications:

- If the *Auxiliary Power PM Enable* bit of the PCIe Device Control register is set, the E610 might consume higher power for any purpose (even if *PME_En* is not set).
- If the *Auxiliary Power PM Enable* bit of the PCIe Device Control register is cleared, higher power consumption is determined by the PCI-PM legacy *PME_En* bit of the PMCSR.

5.2.3 PCIe Link Power Management

The PCIe link state follows the power management state of the device. Since the E610 incorporates multiple PCI functions, the device power management state is defined as the power management state of the most awake function:

- If any function is in D0a state in ARI mode or either D0a or D0u in non-ARI mode, the PCIe link assumes the device is in D0 state.
- Else, if in ARI mode, at least one of the functions is in D3 state and the other functions are not in D0a state, or if in non-ARI mode all of the functions are in the D3 state, the PCIe link assumes the device is in D3 state.
- Else, the device is in Dr state (*PE_RST_N* is asserted to all functions).

The E610 supports the following PCIe power management link states:

- L0 state is used in D0u and D0a states.
- The L1 state is used in D0a and D0u states each time link conditions apply, as well as in the D3 state.
- The L2 state is used in the Dr state following a transition from a D3 state if PCI-PM PME is enabled.
- The L3 state is used in the Dr state following power up, on transition from D0a and also if PME is not enabled in other Dr transitions.

The E610 support for active state link power management is reported via the PCIe Active State Link PM Support field loaded from NVM.

The following NVM fields control L1 behavior:

- Act_Stat_PM_Sup — Indicates support for ASPM L1 in the PCIe configuration space (loaded into the *Active State Link PM Support* field).
- L1_Act_Ext_Latency — Defines L1 active exit latency.
- L1_Act_Acc_Latency — Defines L1 active acceptable exit latency.

5.2.4 Power States

5.2.4.1 D0uninitialized State

The D0u state is a low-power state used after PE_RST_N is de-asserted following power-up (cold or warm), on hot reset (in-band reset through PCIe physical layer message) or on D3 exit. When entering D0u, the E610 disables wake-ups.

Note: Upon exit from D3, if the *I/O Access Enable*, *Memory Access Enable*, or *Bus Lead Enable* bits are set, then the transition is directly to D0active.

5.2.4.1.1 Entry to a D0u State

D0u is reached from either the Dr state (on de-assertion of Internal PE_RST_N) or the D3hot state (by configuration software writing a value of 00b to the *Power State* field of the PCI PM registers).

De-asserting internal PE_RST_N means that the entire state of the device is cleared, other than sticky bits. State is loaded from the NVM, followed by establishment of the PCIe link. Once this is done, configuration software can access the device.

On a transition from D3 to D0u state, the E610 requires that software perform a full re-initialization of the function not including its PCI configuration space which is kept while in D3.

5.2.4.2 D0active State

Once memory space is enabled, the E610 enters an active state. It can transmit and receive packets if properly configured by the software device driver. The PHY is enabled or re-enabled by the software device driver to operate/auto-negotiate to full line speed/power if not already operating at full capability. Any APM wake-up previously active remains active. The software device driver can deactivate ACPI wake-up by writing to the Wake-Up Control (WUC) register and APM wake-up by writing to the General Receive Control (GRC) register, or activate other wake-up filters by writing to the Wake-Up Filter Control (WUFC) register.

5.2.4.2.1 Entry to D0a State

D0a is entered from the D0u state by writing a 1b to the *Memory Access Enable* or the *I/O Access Enable* bit of the PCI Command register. The DMA, MAC, and PHY of the appropriate LAN function are enabled.

5.2.4.3 D3 State (PCI-PM D3hot)

The E610 transitions to D3 when the system writes a 11b to the *Power State* field of the PMCSR. Any wake-up filter settings that were enabled before entering this reset state are maintained. Upon transition to D3 state, the E610 clears the *Memory Access Enable* and *I/O Access Enable* bits of the PCI Command register, which disables memory access decode. In D3 the E610 only responds to PCI configuration accesses and does not generate Lead cycles.

Configuration and message requests are the only PCIe TLPs accepted by a function in the D3hot state. All other received requests must be handled as unsupported requests, and all received completions can optionally be handled as unexpected completions. If an error caused by a received TLP (like an unsupported request) is detected while in D3hot, and reporting is enabled, the link must be returned to L0 if it is not already in L0 and an error message must be sent. See section 5.3.1.4.1 in the PCIe Base Specification.

A D3 state is followed by either a D0u state (in preparation for a D0a state) or by a transition to Dr state (PCI-PM D3cold state). To transition back to D0u, the system writes a 00b to the *Power State* field of the PMCSR. Transition to Dr state is through PE_RST_N assertion.

5.2.4.3.1 Entry to D3 State

Transition to D3 state is through a configuration write to the *Power State* field of the PCI-PM registers.

Prior to transition from D0 to the D3 state, the software device driver disables scheduling of further tasks to the E610; it masks all interrupts, it does not write to the Transmit Descriptor Tail (TDT) register or to the Receive Descriptor Tail (RDT) register and operates the bus Lead disable algorithm as defined in [Section 5.2.4.3.2](#). If wake-up capability is needed, the software device driver should set up the appropriate wake-up registers and the system should write a 1b to the *PME_En* bit of the PMCSR prior to the transition to D3.

If all PCI functions are programmed into D3 state, the E610 brings its PCIe link into the L1 link state. As part of the transition into L1 state, the E610 suspends scheduling of new TLPs and waits for the completion of all previous TLPs it has sent. The E610 clears the *Memory Access Enable* and *I/O Access Enable* bits of the PCI Command register, which disables memory access decode. Any receive packets that have not been transferred into system memory is kept in the device (and discarded later on D3 exit). Any transmit packets that have not been sent can still be transmitted (assuming the Ethernet link is up).

In preparation to a possible transition to D3cold state, and to reduce power consumption, whenever entering D3 state, the software device driver might disable one of the LAN ports (LAN disable) and/or the PHY(s) auto-negotiate network link(s) to a lower speed (if supported by the network interface). See [Section 5.3.5.2](#) for a description of network interface behavior in this case.

5.2.4.3.2 Bus Lead Disable

System software enables Lead accesses on the PCIe link by setting the *Bus Lead Enable* bit (BME) in the PCIe Command Register. System software might disable Lead accesses on the PCIe link by either clearing the *BME* bit or by bringing the function into a D3 state. From that time on, the E610 must not issue Lead accesses for this function. Due to the full-duplex nature of PCIe, and the pipelined design in the E610, it might happen that multiple requests from several functions are pending when the Lead disable request arrives. The protocol described in this section insures that a function does not issue Lead requests to the PCIe link after its *Lead Enable* bit is cleared (or after entry to D3 state).

Two configuration bits are provided for the handshake between the device function and its software device driver:

- *PCIE_Lead_Disable* bit in the Device Control Register (CTRL) register — When the *PCIE_Lead_Disable* bit is set, the E610 blocks new Lead requests by this function. The E610 then proceeds to issue any pending requests by this function. This bit is cleared on Lead reset (LAN_PWR_GOOD all the way to software reset) to allow Lead accesses.
- *PCIE_Lead_Enable_Status* bits in the Device Status register — Cleared by the E610 when the *PCIE_Lead_Disable* bit is set and no Lead requests are pending by the relevant function. Set otherwise. Indicates that no Lead requests are issued by this function as long as the *PCIE_Lead_Disable* bit is set. The following activities must end before the E610 clears the *PCIE_Lead_Enable_Status* bit:

Notes: The software device driver is expected to disable any reception to the Rx queues as described in [Section 4.6.7.1.1](#). Then, the software device driver sets the *PCIE_Lead_Disable* bit when notified of a pending Lead disable (or D3 entry). The E610 then blocks new requests and proceeds to issue any pending requests by this function. The software device driver then reads the change made in the *PCIE_Lead_Disable* bit and then polls the *PCIE_Lead_Enable_Status* bit. Once the bit is cleared, it is guaranteed that no requests are pending from this function.

The software device driver might time out if the *PCIE_Lead_Enable_Status* bit is not cleared within a given time. Examples for cases that the E610 might not clear the *PCIE_Lead_Enable_Status* bit for a long time are cases of flow control, link down, or DMA completions not making it back to the DMA block. In these cases, the software device driver should check that the *Transaction Pending* bit (bit 5) in the Device Status register in the PCI config space is cleared before proceeding. In such cases, the driver needs to clear internal buffers by:

- Setting the HLREG0.LPBK bit.
- Clearing RXCTRL.RXEN bit.
- Setting the setting GCR_EXT.Buffers_Clear_Func bit for 20 μ s.
- Initiate two consecutive software resets with a delay larger than 1 μ s between them.

Intel's device driver software times-out at ~ 800 μ s. The *PCIE_Lead_Disable* bit must be cleared to enable Lead request to the PCIe link. This bit should be cleared through reset.

5.2.4.4 Dr State

Transition to Dr state is initiated on several cases:

- **On system power up** — Dr state begins with the assertion of the internal power detection circuit (LAN_PWR_GOOD) and ends with de-assertion of PE_RST_N.
- **On transition from a D0a state** — During operation the system might assert PE_RST_N at any time. In an ACPI system, a system transition to the G2/S5 state causes a transition from D0a to Dr state.
- **On transition from a D3 state** — The system transitions the device into the Dr state by asserting PCIe PE_RST_N.

Any wake-up filter settings that were enabled before entering this reset state are maintained.

The system might maintain PE_RST_N asserted for an arbitrary time. The de-assertion (rising edge) of PE_RST_N causes a transition to D0u state.

While in Dr state, the E610 might maintain functionality (for WoL or manageability) or might enter a Dr Disable state (if no WoL and no manageability) for minimal device power. The Dr Disable mode is described in the sections that follow.

5.2.4.4.1 Dr Disable Mode

The E610 enters a Dr disable mode on transition to D3cold state when it does not need to maintain any functionality. The conditions to enter either state are:

- All PCI functions are in the Dr state.
- APM WoL is inactive for both LAN functions.
- Pass-through manageability is disabled.
- ACPI PME is disabled for all PCI functions.

Entry into Dr disable is usually done on assertion of PCIe PE_RST_N. It might also be possible to enter Dr disable mode by reading the NVM while already in Dr state. The usage model for this later case is on system power up, assuming that manageability and wake-up are not required. Once the E610 enters Dr state on power-up, the NVM is read. If the NVM contents determine that the conditions to enter Dr Disable are met, the E610 then enters this mode (assuming that PCIe PE_RST_N is still asserted).

Exit from Dr disable is through de-assertion of PCIe PE_RST_N.

If Dr disable mode is entered from D3 state, the platform might remove E610 power. If the platform removes E610 power, it must remove all power rails from the E610. Exit from this state is through power-up cycle to the E610.

5.2.4.4.2 Entry to Dr State

Dr-entry on platform power-up is as follows:

- Asserting the internal power detection circuit (LAN_PWR_GOOD). Device power is kept to a minimum by keeping the PHYs in low power.
- The NVM is then read and determines device configuration.
- If the *APM Enable* bit in *NVM Control Word 3* is set, APM wake-up is enabled (for each port independently).
- Each of the LAN ports can be enabled if required for WoL or manageability. See [Section 5.3.5.2](#) for exact condition to enable a port. In such a case, the PHY might auto-negotiate to a lower speed on Dr entry (see [Section 5.3.5](#)).
- The PCIe link is not enabled in Dr state following system power up (since PE_RST_N is asserted).

Entry to Dr state from D0a state is by asserting the PE_RST_N signal. An ACPI transition to the G2/S5 state is reflected in a device transition from D0a to Dr state. The transition can be orderly (such as user selected a shut down operating system option), in which case the software device driver might have a chance to intervene. Or, it might be an emergency transition (like power button override), in which case, the software device driver is not notified.

To reduce power consumption, if any of manageability, APM wake or PCI-PM PME¹ is enabled, the PHY auto-negotiates to a lower link speed on D0a to Dr transition (see [Section 5.3.5](#)).

1. ACPI 2.0 specifies that OSPM does not disable wake events before setting the *SLP_EN* bit when entering the S5 sleeping state. This provides support for remote management initiatives by enabling Remote Power On (RPO) capability. This is a change from ACPI 1.0 behavior.

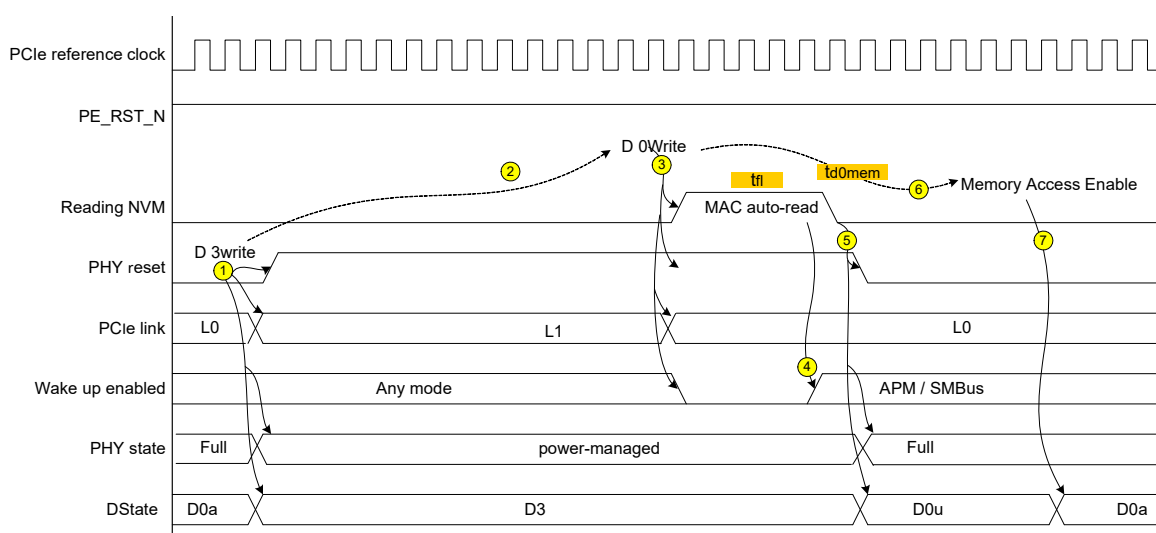
Transition from D3 state to D0 state is done by asserting the PE_RST_N signal. Prior to that, the system initiates a transition of the PCIe link from L1 state to either the L2 or L3 state (assuming all functions were already in D3 state). The link enters L2 state if PCI-PM PME is enabled.

5.2.5 Timing of Power-State Transitions

The following sections give detailed timing for the state transitions. In the diagrams, the dotted connecting lines represent the E610 requirements, while the solid connecting lines represent the E610 guarantees.

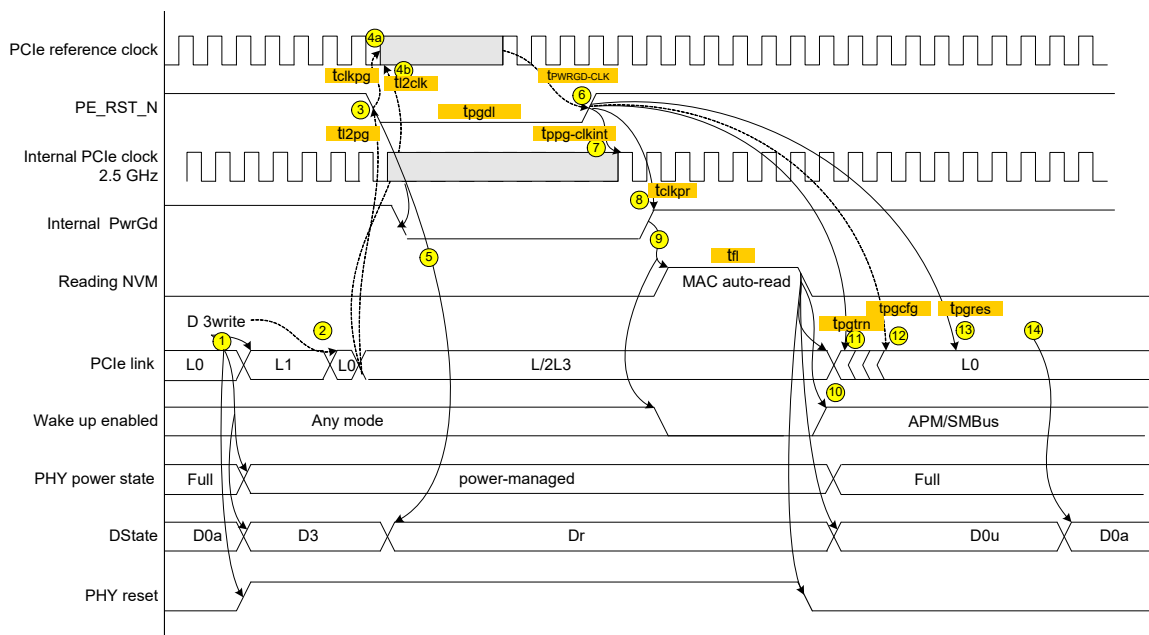
Note that the timing diagrams are not to scale. The clock's edges are shown to indicate running clocks only and are not used to indicate the actual number of cycles for any operation.

5.2.5.1 Transition from D0a to D3 and Back without PE_RST_N



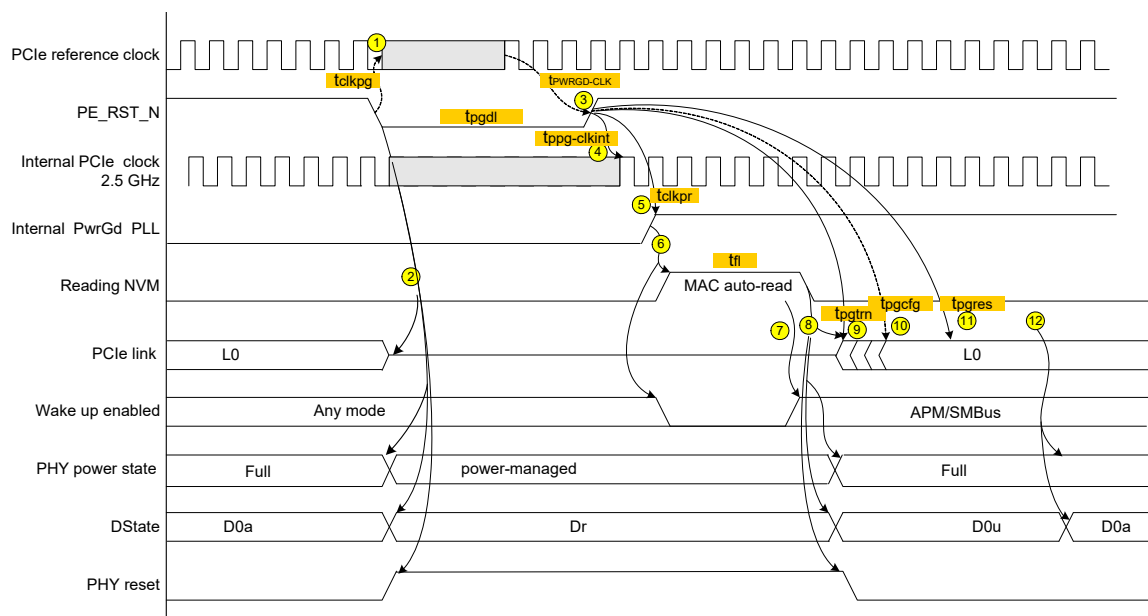
No.	Notes
1	Writing 11b to the <i>Power State</i> field of the PMCSR transitions the E610 to D3. PHY re-negotiates to a lower link speed once <i>VE TO</i> bit is cleared if manageability and/or WoL is enabled, else it is powered down. When in power-down mode, PHY is reset and its CSRs are reset to their default values according to provisional register segment loaded from NVM at last LAN_PWR_GOOD event.
2	The system can keep the E610 in D3 state for an arbitrary amount of time.
3	To exit D3 state the system writes 00b to the <i>Power State</i> field of the PMCSR.
4	APM wake-up or manageability might be enabled based on what is read in the NVM.
5	After reading the MAC section of the NVM, the LAN ports are enabled and the E610 transitions to D0u state. PHY re-negotiates to full link speed.
6	The system can delay an arbitrary time before enabling memory access.
7	Writing a 1b to the <i>Memory Access Enable</i> bit or to the <i>I/O Access Enable</i> bit in the PCI Command register transitions the E610 from D0u to D0 state.

5.2.5.2 Transition from D0a to D3 and Back with PE_RST_N



No.	Notes
1	Writing 11b to the <i>Power State</i> field of the PMCSR transitions the E610 to D3. PCIe link transitions to L1 state. PHY re-negotiates to a lower link speed once <i>VETO</i> bit is cleared if manageability and/or WoL is enabled, else it is powered down. When in power-down mode, PHY is reset and its CSRs are reset to their default values according to provisional register segment loaded from NVM at last LAN_PWR_GOOD event.
2	The system can delay an arbitrary amount of time between setting D3 mode and transitioning the link to an L2 or L3 state.
3	Following link transition, PE_RST_N is asserted.
4	The system must assert PE_RST_N before stopping the PCIe reference clock. It must also wait t_{l2clk} after link transition to L2/L3 before stopping the reference clock.
5	On assertion of PE_RST_N, the E610 transitions to Dr state. Once entering Dr state, if manageability and/or WoL is enabled, PHY re-negotiates a lower link speed in any case, even if the <i>VETO</i> bit is asserted, else it is maintained powered down.
6	The system starts the PCIe reference clock $t_{PWRGD-CLK}$ before de-assertion PE_RST_N.
7	The Internal PCIe clock is valid and stable $t_{ppg-clkint}$ from PE_RST_N de-assertion.
8	The PCIe Internal PWRGD signal is asserted t_{clkpr} after the external PE_RST_N signal.
9	Assertion of Internal PCIe PWRGD causes the MAC section in the NVM to be re-read and disables wake-up. PHY is reset only if manageability unit and WoL are disabled. In such a case, PHY CSRs are reset to their default values according to provisional register segment loaded from NVM on last power-up event.
10	APM wake-up mode might be enabled based on what is read from the NVM. PHY re-negotiates to full link speed.
11	Link training starts after t_{pgtrn} from PE_RST_N de-assertion.
12	A first PCIe configuration access might arrive after t_{pgcfg} from PE_RST_N de-assertion.
13	A first PCI configuration response can be sent after t_{pgres} from PE_RST_N de-assertion.
14	Writing a 1b to the <i>Memory Access Enable</i> bit in the PCI Command register transitions the device from D0u to D0 state.

5.2.5.3 Transition from D0a to Dr and Back without Transition to D3



No.	Notes
1	The system must assert PE_RST_N before stopping the PCIe reference clock. It must also wait t12clk after link transition to L2/L3 before stopping the reference clock.
2	On assertion of PE_RST_N, the E610 transitions to Dr state and the PCIe link transition to electrical idle. Once entering Dr state, if manageability and/or WoL is enabled, PHY re-negotiates a lower link speed in any case, even if the VETO bit is asserted, else it is powered down. When in power-down mode, PHY is reset and its CSRs are reset to their default values according to provisional register segment loaded from NVM at last LAN_PWR_GOOD event
3	The system starts the PCIe reference clock t _{PWRGD-CLK} before de-assertion PE_RST_N.
4	The Internal PCIe clock is valid and stable t _{pgg-clkint} from PE_RST_N de-assertion.
5	The PCIe Internal PWRGD signal is asserted t _{clkpr} after the external PE_RST_N signal.
6	Asserting Internal PCIe PWRGD causes the MAC section in the NVM to be re-read and disables wake-up. PHY is reset only if manageability unit and WoL are disabled. In such a case, PHY CSRs are reset to their default values according to provisional register segment loaded from NVM on last power-up event.
7	APM wake-up mode might be enabled based on what is read from the NVM.
8	After reading the NVM, PHY re-negotiates to full link speed.
9	Link training starts after t _{pgtrn} from PE_RST_N de-assertion.
10	A first PCIe configuration access might arrive after t _{pgcfg} from PE_RST_N de-assertion.
11	A first PCI configuration response can be sent after t _{pgres} from PE_RST_N de-assertion.
12	Writing a 1b to the <i>Memory Access Enable</i> bit in the PCI Command register transitions the device from D0u to D0 state.

5.2.5.4 Timing Requirements

The E610 requires the following start-up and power-state transitions.

Table 5-1. Start-up and Power-State Transitions

Parameter	Description	Min	Max	Notes
t_{xog}	Xosc stable from power stable.		10 ms	
$t_{PWRGD-CLK}$	PCIe clock valid to PCIe power good.	100 μ s	-	According to PCIe specification.
t_{pVPG}	Power rails stable to PCIe PE_RST_N inactive.	100 ms	-	According to PCIe specification.
t_{pgcf}	External PE_RST_N signal to first configuration cycle.	100 ms		According to PCIe specification.
t_{d0mem}	Device programmed from D3h to D0 state to next device access.	10 ms		According to PCI power management specification.
t_{l2pg}	L2 link transition to PE_RST_N assertion.	0 ns		According to PCIe specification.
t_{l2clk}	L2 link transition to removal of PCIe reference clock.	100 ns		According to PCIe specification.
t_{clkpg}	PE_RST_N assertion to removal of PCIe reference clock.	0 ns		According to PCIe specification.
t_{pgd}	PE_RST_N assertion time.	100 μ s		According to PCIe specification.

5.2.5.5 Timing Guarantees

The E610 guarantees the following start-up and power state transition related timing parameters.

Table 5-2. Start-up and Power-State Transition Timing Parameters

Parameter	Description	Min	Max	Notes
t_{xog}	Xosc stable from power stable.		10 ms	
t_{ppg}	Internal power good delay from valid power rail.	35 ms	35 ms	
t_{fl}	NVM read duration.		20 ms	
$t_{ppg-clkint}$	PCIe PE_RST_N to internal PLL lock.	-	50 μ s	
t_{clkpr}	Internal PCIe PWGD from external PCIe PE_RST_N.		50 μ s	
t_{pgtrn}	PCIe PE_RST_N to start of link training.		20 ms	According to PCIe specification.
t_{pgres}	External PE_RST_N to response to first configuration cycle.		1 second	According to PCIe specification.

5.3 Network Interfaces Power Management

The E610 transits any of the network interfaces into a low-power state in the following cases:

- The respective LAN function is in LAN disable mode if the DEV_DIS_N pin is asserted to low and the ALLOW_PORT_DIS flag of the port in the NVM is active.
- The E610 is in D3 or Dr state, APM WoL is disabled for the port, ACPI wake is disabled for the port and pass-through manageability is disabled for the port.

Use of the LAN ports for pass-through manageability follows the following behavior:

- If manageability is disabled (*Manageability Pass Through Mode* field in the Common Manageability Parameters NVM word is set to *None* - [Section 6.3.49.2](#)), LAN ports are not allocated for manageability.
- If manageability is enabled:
 - Power-up — Following an NVM read, a single port is enabled for manageability, running at the lowest speed supported by the interface. If APM WoL is enabled on a single port, the same port is used for manageability. Otherwise, manageability protocols (such as teaming) determine which port is used.
 - D0 state — Both LAN ports are enabled for manageability.
 - D3 and Dr states — A single port is enabled for manageability, running at the lowest speed supported by the interface. If WoL is enabled on a single port, the same port is used for manageability. Otherwise, manageability protocols such as teaming) determine which port is used.

Enabling a port as a result of the previous reasons cause an internal reset of the port including its associated PHY.

When a network interface is in low-power state, the E610 MAC asserts internal signals to notify the PHY that it must either power down as well or re-negotiate to a lower link speed.

5.3.1 PHY Power-Down State

Each E610 port enters a power-down state when none of its clients is enabled and therefore has no need to maintain a link. This can happen in one of the following cases:

1. **D3/Dr state** — Each PHY enters a low-power state if all the following conditions are met:
 - a. The LAN function associated with this PHY is in a non-D0 state
 - b. APM WoL is inactive
 - c. Manageability does not use this port
 - d. ACPI PME is disabled for this port
2. **LAN disable pin** — Each PHY is disabled if the associated LAN disable input pin indicates that the port should be disabled.
3. **LAN PCI Disable bit in NVM** — A single LAN port can also be disabled through NVM settings. If the *LAN PCI Disable* bit is set in *NVM Control Word 2*, and if the port is not used for WoL or for MC traffic, the *LAN Disable Select* bit selects the MAC and PHY port that enters power down even in D0 state. Note that if the port is used for WoL or by the MC, setting the *LAN PCI Disable* bit in *NVM Control Word 2* does not bring the MAC and PHY into power down, but only the DMA block.

When powered down by one of these means, a significant portion of the MAC and the PHY (including its microprocessor, MDIO, and interrupt logic) are powered down. Only the PHY PLLs are functioning, and the Analog Front-End (AFE) is turned off (gets placed in high-impedance mode).

When the E610 is completely powered down (Dr state), the PHYs reach a deeper power saving mode.

When the PHY exits power down, it re-initializes all analog functions, and retrieves the default configuration settings that were loaded from NVM the last time NVM was read.

5.3.2 PHY Power-Down via the PHY Register

The PHY can be powered down by setting a *Power Down* bit in a PHY register (Register 0_0.11 for BASE-T mode or register 0_1.11 for Fiber mode). This bit powers down a significant portion of the PHY but clocks provided to the MAC and to the register section of the PHY remain active. Only the MDIO, interrupts, and microprocessor are functioning, and the AFE is turned off (gets placed in high-impedance mode).

When the PHY exits software power down (*Power Down* bit cleared), it re-initializes all analog functions, but retains its previous configuration settings.

5.3.3 Smart Power-Down (SPD)

If the link is lost, the embedded firmware can put the port into the SPD mode by accessing the PHY registers. The SPD modes are enabled only for a copper link and enabled Link Auto-negotiation. The PHY can be placed in one of two SPD modes.

- SPD Mode 1 (Detect Link)
- SPD Mode 2 (Detect Link + Deadlock Avoidance)

SPD Mode 1 (Detect Link):

SPD Mode 2 (Detect Link + Deadlock Avoidance):

5.3.4 Disable 10GBASE-T and/or 1000BASE-T Speeds

To reduce power consumption of a port even when it is active, there is an option to disable 10 Gb/s and/or 1000 Mb/s advertisement during auto-negotiation.

This can be useful for cases where a copper link is used as a dormant link for redundancy purposes only.

Note: These bits disable the advertisement of a speed in any power state, whether the port is active (D0u or D0a) or in low power (Dr or D3).

5.3.5 Energy Efficient Ethernet (EEE)

the E610 supports Energy Efficient Ethernet when connected to copper PHY that supports it. the E610 enters EEE Low Power Idle (LPI) mode on transmit or receive independently each time the E610 detects no data is scheduled for transmission, or the link partner indicates no data is pending for reception.

EEE LPI mode defined in IEEE802.3az enables power saving by switching off part of the E610 functionality when no data needs to be transmitted or/and received. Decision on whether the E610 transmit path should enter LPI mode or exit LPI mode is done according to a need to transmit. Information on whether a link partner has entered LPI mode is detected by the E610 and utilized for power saving in the receive circuitry.

When no data needs to be transmitted, a request to enter transmit LPI is issued on the internal xxMII Tx interface causing the PHY to transmit sleep symbols for a predefined period of time followed by a quiet period. During LPI, the PHY periodically transmits refresh symbols that are used by the link partner to update adaptive filters and timing circuits to maintain link integrity. This quiet-refresh cycle continues until transmitting 'normal inter-frame' encoding on the internal xxMII Tx interface. The PHY communicates to the link partner the move to link active state by sending wake symbols for a predefined period of time. The PHY then enters a normal operating state where data or idle symbols are transmitted.

In the receive direction, entering LPI mode is triggered by receiving sleep symbols from the link partner. This signals that the link partner is about to enter LPI mode. After sending the sleep symbols, the link partner ceases transmission. When the link partner initiates the move to LPI, the PHY indicates "assert low power idle" on the internal xxMII Rx interface and the E610 receiver disables functionality to reduce power consumption.

Figure 5-2 and Table 5-3 illustrate the general principles of EEE LPI operation on the Ethernet link.

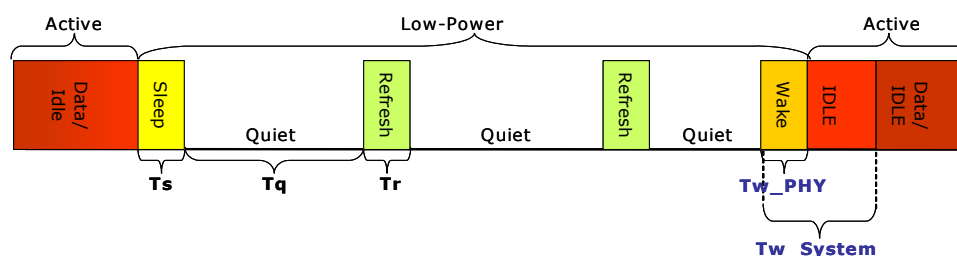


Figure 5-2. EEE Operation

Table 5-3. EEE Parameters

Parameter	Description
Sleep Time (Ts)	Duration PHY sends sleep symbols before going quiet.
Quiet Duration (Tq)	Duration PHY remains quiet before it must wake for refresh period.
Refresh Duration (Tr)	Duration PHY sends refresh symbols for timing recovery and coefficient synchronization.
PHY Wake Time (Tw_PHY)	Minimum duration PHY takes to resume to active state after decision to wake.
Receive System Wake Time (Tw_System_rx)	Wait period where no data is expected to be received to give the local receiving system time to wake-up.
Transmit System Wake Time (Tw_System_tx)	Wait period where no data is transmitted to give the remote receiving system time to wake-up.

5.3.5.1 Conditions to Enter EEE Tx LPI

In the transmit direction entry into EEE LPI mode of operation is triggered when one of the following conditions exist:

1. No transmission is pending, management does not need to transmit and internal transmit buffer is empty and the `EEER.TX_LPI_EN` bit is set to 1b.
2. If the `EEER.TX_LPI_EN` bit is set to 1b and a XOFF flow control packet is received from the link partner or PFC XOFF are received for all the traffic classes supporting PFC, the E610 moves the link into the Tx LPI state for the pause duration even if a transmission is pending.

When one of the previous conditions to enter Tx LPI state are detected “assert LPI” is transmitted on the internal xxMII interface and the E610 PHY transmits sleep symbols on the network interface to communicate to the link partner entry into Tx LPI link state. After sleep symbols transmission, the PHY immediately enters the low power quiet mode. In this state the PHY periodically transitions between quiet link state, where link is idle, to sending refresh symbols until a request to transition link back to normal (active) mode is transmitted on the internal xxMII Tx interface (See [Figure 5-2](#)).

Note: Initial `EEER.TX_LPI_EN` configuration is loaded from NVM.

Note: 10GBASE-T Ethernet LPI allows each link direction to enter sleep, refresh or wake states asymmetric from the other direction.

Note: EEE LPI status of a E610 port can be found in the `EEE_STAT` register.

5.3.5.2 Transition from Tx LPI to Active Link State

The E610 exits Tx LPI link state and transition link into active link state when none of the conditions defined in [Section 5.3.5.1](#) exist. To transition into active link state, the E610 transmits:

1. Normal ‘inter-frame’ encoding on the internal xxMII Tx interface for a pre-defined link rate dependent period time of `Tw_sys_tx-min` (defined by the `EEE_SU` register and IEEE802.3az clause 78.5). As a result, the PHY transmits wake symbols for a `Tw_phy` duration followed by idle symbols.
2. If the `Tw_System_tx` duration defined in the `EEER.TW_SYSTEM` field is longer than `Tw_sys_tx-min`, the E610 continues transmitting the ‘inter-frame’ encoding on the internal xxMII interface until the time defined in the `EEER.TW_SYSTEM` field has expired, before transmitting the actual data. During this period the PHY continues transmitting idle symbols.

Note: When moving out of Tx LPI to transmit a 802.3x flow control frame, the E610 waits for the `Tw_sys_tx-min` duration before transmitting the flow control frame. It should be noted that even in this scenario actual data is transmitted only after the `Tw_System_tx` time defined in the `EEER.TW_SYSTEM` field has expired.

5.3.5.3 EEE Auto-Negotiation

Auto-negotiation provides the capability to negotiate EEE capabilities with the link partner using the next page mechanism defined in IEEE802.3 Annex 73A. IEEE802.3 auto-negotiation is performed at power up, on command from software, upon detection of a PHY error or following link re-connection.

During the link establishment process, both link partners indicate their EEE capabilities using IEEE802.3 auto-negotiation. If EEE is supported by both link partners for the negotiated PHY type, the EEE function can be used independently in either direction.

5.3.5.4 EEE Link Level (LLDP) Capabilities Discovery

The E610 supports LLDP negotiation via software using the EEE IEEE802.1AB Link Layer Discovery Protocol (LLDP) Type, Length, Value (TLV) fields defined in IEEE802.3az clause 78 and clause 79. LLDP negotiation enables negotiation of increased system wake time (Transmit T_w and Receive T_w) to enable improving system energy efficiency.

5.3.5.4.1 LLDP Negotiation Actions

Following negotiation of a new system wake time via EEE LLDP negotiation, the following fields and registers should be updated:

The `EEER.TW_SYSTEM` field with the negotiated Transmit T_w time value, to increase the duration where idle symbols are transmitted following a move out of EEE Tx LPI state before actual data can be transmitted.

- A value placed in `EEER.TW_SYSTEM` field does not effect transmission of flow control packets. Depending on the technology flow control packet, transmission is delayed following a move out of EEE Tx LPI state only by the minimum `Tw_sys_tx` time as defined in IEEE802.3az clause 78.5. In the E610 the minimum `Tw_sys_tx` time value is defined in the `EEE_SU` register together with time defined in IEEE802.3az clause 78.5. Value varies as a function of link rate and technology.

5.3.5.5 EEE Statistics

The E610 supports reporting a number of EEE LPI Tx and Rx events via the RLPIC and TLPIC registers.

5.4 Wake-Up

5.4.1 Advanced Power Management Wake-Up

Advanced power management wake-up, or APM wake-up, was previously known as Wake on LAN (WoL). It is a feature that has existed in the 10/100 Mb/s NICs for several generations. The basic premise is to receive a packet with an explicit data pattern, and then to assert a signal to wake-up the system. In the earlier generations, this was accomplished by using a special signal that ran across a cable to a defined connector on the motherboard. The NIC asserts the signal for approximately 50 ms to signal a wake-up. The E610 uses (if configured to) an in-band PM_PME message for this (if the PCIe is inactive then a PE_WAKE_N signal is asserted first to resume power and clock to the PCIe interface).

At power up, the E610 reads the *APM Enable* bit from the NVM into the *APM Enable* (APME) bits of the GRC register. This bit control the enabling of *APM Wakeup*.

When *APM Wakeup* is enabled and when not in D0 state, the E610 checks all incoming packets for Magic Packets. See [Section A.1.3](#) for a definition of Magic Packets.

Once the E610 receives a matching magic packet, it:

- Sets the *PME_Status* bit in the *PMCSR* and issues a PM_PME message (in some cases, this might require to assert the PE_WAKE_N signal first to resume power and clock to the PCIe interface).

Note: At wake-up from first power on, the PM_PME message is not sent.

- Stores the first 128 bytes of the packet in the Wake-Up Packet Memory (WUPM).
- Sets the *Magic Packet Received* bit (*MAG*) in the Wake-Up Status (WUS) register.
- Sets the packet length in the Wake-Up Packet Length Register (WUPL).

The E610 maintains the first magic packet received in the Wake-Up Packet Memory (WUPM) until the software device driver writes a 0b to the Magic Packet Received *MAG* bit in the Wake-Up Status Register (WUS), hence the software device driver has to read the packet from WUPM prior to clearing the WUS indication.

APM Wakeup event is issued only when in Dx power states and it is disabled only if a subsequent NVM read results in the *APM Wake-Up* bit being cleared or if the software explicitly writes a 0b to the *APM Wake-Up* (APM) bit of the GRC register.

5.4.2 ACPI Power Management Wake-Up

The E610 supports ACPI power management based wake-up. It can generate system wake-up events from three sources, regardless to the power state:

- Receiving a Magic Packet.
- Receiving a network wake-up packet.
- Detecting a link change of state.
- Firmware reset event.

Activating ACPI power management wake-up requires the following steps:

- The operating system (at configuration time) writes a 1b to the *PME_En* bit of the PMCSR (PMCSR.8).
- The software device driver clears all pending wake-up status in the WUS register by writing 1b to all the status bits.
- The software device driver programs the WUFC register to indicate the packets it needs to wake-up and supplies the necessary data to the IPv4/v6 Address Table (IP4AT, IP6AT), Flexible Host Filter Table (FHFT). It can also set the *Link Status Change Wake-Up Enable (LNKC)* bit in the WUFC register to cause wake-up when the link changes state.
- Once the E610 wakes the system the software device driver needs to clear the WUS and WUFC registers until the next time the system goes to a low power state with wake-up.

Normally, after enabling ACPI wake-up, the operating system writes (11b) to the lower two bits of the PMCSR to put the E610 into low-power mode, and once entering D0, OS disables ACPI wake-up.

Once wake-up is enabled, the E610 monitors incoming packets, first filtering them according to its standard address filtering method, then filtering them with all of the enabled wake-up filters. If a packet passes both the standard address filtering and at least one of the enabled wake-up filters, the E610:

- Sets the *PME_Status* bit in the PMCSR.
- If the *PME_En* bit in the PMCSR is set and device is in Dr state or *PCI_GLBL_CNF.WAKE_PIN_EN* is set in all states, asserts *PE_WAKE_N*.
- In non Dr state or when exiting Dr, a *PM_PME* message is also issued.
- Stores the first 128 bytes of the packet in the Wake-Up Packet Memory (WUPM) register. Sets one or more of the *Received* bits in the WUS register. Note that the E610 sets more than one bit if a packet matches more than one filter. Sets the packet length in the Wake-Up Packet Length Register (WUPL).

If enabled, a link state change wake-up causes similar results, setting *PME_Status*, asserting *PE_WAKE_N* and setting the *LNKC* bit in the WUS register when the link goes up or down.

Note: As the LPLU mechanism may cause a link change when entering Sx, setting the link state change wake-up while link is up may cause an immediate wake-up when entering Sx. If this is needed, the software device driver should move the link to the lowest available link speed before enabling the wake-up event and entering Sx.

If enabled, a firmware reset causes similar results, setting *PME_Status*, asserting *PE_WAKE_N* and setting the *FW_RST_WK* bit in the WUS register when the firmware is reset.

PE_WAKE_N remains asserted until the operating system either writes a 1b to the *PME_Status* bit of the PMCSR register or writes a 0b to the *PME_En* bit.

5.4.3 Wake-Up Packets

The E610 supports various wake-up packets using two types of filters:

- Pre-defined filters
- Flexible filters

Each of these filters are enabled if the corresponding bit in the WUFC register is set to 1b.

5.4.3.1 Pre-Defined Filters

The following packets are supported by the E610's pre-defined filters:

- Directed packet (including exact, multicast indexed, and broadcast)
- Magic Packet
- ARP/IPv4 request packet
- Directed IPv4 packet
- Directed IPv6 packet

Each of these filters are enabled if the corresponding bit in the WUFC register is set to 1b.

The description of each filter includes a table showing which bytes at which offsets are compared to determine if the packet passes the filter. Both VLAN frames and LLC/SNAP can increase the given offsets if they are present.

5.4.3.1.1 Directed Packets

Unicast — The E610 generates a wake-up event after receiving any packet whose destination address matches one of the 128 valid programmed receive addresses if the *Directed Exact Wake Up Enable* bit is set in the WUFC register (WUFC.EX).

For multicast packets, the upper bits of the incoming packet's destination address index a bit vector, the Multicast Table Array (MTA) that indicates whether to accept the packet. If the *Directed Multicast Wake Up Enable* bit set in the WUFC register (WUFC.MC) and the indexed bit in the vector is one, the E610 generates a wake-up event. The exact bits used in the comparison are programmed by software in the *Multicast Offset* field of the Multicast Control register (MCSTCTRL.MO).

If the *Broadcast Wake Up Enable* bit in the WUFC register (WUFC.BC) is set, the E610 generates a wake-up event when it receives a broadcast packet.

5.4.3.1.2 Magic Packet

See [Section A.1.3](#) for a definition of Magic Packets.

A Magic Packet's destination address must match the address filtering enabled in the configuration registers with the exception that broadcast packets are considered to match even if the *Broadcast Accept* bit of the Receive Control register (FCTRL.BAM) is 0b. If *APM Wakeup* is enabled in the NVM, the E610 starts up with the Receive Address Register 0 (RAH0, RAL0) loaded from the NVM. This enables the E610 to accept packets with the matching IEEE address before the software device driver comes up.

Note: Setting the FCTRL.MPE does not allow reception of magic packets with a multicast address. To receive such packets, the multicast address should be stored in an RHA/RAL register or enabled in the MTA table.

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Compare	MAC header – processed by main address filter
6	6	Source Address		Skip	
12	S= (0/4)	Possible VLAN Tag		Skip	
12/16	D=(0/8)	Possible Len/LLC/SNAP Header		Skip	
12+S+D	2	Type		Skip	
Any	6	Synchronizing Stream	FF*6+	Compare	
any+6	96	16 Copies of Node Address	A*16	Compare	Compared to Receive Address Register 0 (RAH0, RAL0)

5.4.3.1.3 ARP/IPv4 Request Packet

The E610 supports receiving ARP request packets for wake-up if the *ARP* bit is set in the WUFC register. Four IPv4 addresses are supported, which are programmed in the IPv4 Address Table (IP4AT). A successfully matched packet must pass L2 address filtering, a Protocol Type of 0x0806, an ARP OpCode of 0x01, and one of the four programmed IPv4 addresses. The E610 also handles ARP request packets that have VLAN tagging on both Ethernet II and Ethernet SNAP types.

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Compare	MAC header – processed by main address filter
6	6	Source Address		Skip	
12	S=(0/4)	Possible VLAN Tag		Compare	
12/16	D=(0/8)	Possible Len/LLC/SNAP Header		Skip	
12+S+D	2	Type	0x0806	Compare	ARP
14+S+D	2	Hardware Type	0x0001	Compare	
16+S+D	2	Protocol Type	0x0800	Compare	
18+S+D	1	Hardware Size	0x06	Compare	
19+S+D	1	Protocol Address Length	0x04	Compare	
20+S+D	2	Operation	0x0001	Compare	
22+S+D	6	Sender Hardware Address	-	Ignore	
28+S+D	4	Sender IP Address	-	Ignore	
32+S+D	6	Target Hardware Address	-	Ignore	
38+S+D	4	Target IP Address	IP4AT	Compare	Can match any of 4 values in IP4AT

5.4.3.1.4 Directed IPv4 Packet

The E610 supports receiving directed IPv4 packets for wake-up if the *IPv4* bit is set in the WUFC register. Four IPv4 addresses are supported that are programmed in the IPv4 Address Table (IP4AT). A successfully matched packet must pass L2 address filtering, a Protocol Type of 0x0800, and one of the four programmed IPv4 addresses. The E610 also handles directed IPv4 packets that have VLAN tagging on both Ethernet II and Ethernet SNAP types.

5.4.3.1.5 Directed IPv6 Packet

The E610 supports receiving directed IPv6 packets for wake-up if the *IPV6* bit is set in the WUFC register. One IPv6 address is supported and it is programmed in the IPv6 Address Table (IP6AT). A successfully matched packet must pass L2 address filtering, a Protocol Type of 0x0800, and the programmed IPv6 address. The E610 also handles directed IPv6 packets that have VLAN tagging on both Ethernet II and Ethernet SNAP types.

5.4.3.2 Flexible Filter

The E610 supports a total of **eight** host flexible filters. Each filter can be configured to recognize any arbitrary pattern within the first 128 bytes of the packet. To configure the flexible filter, software programs the required values into the Flexible Host Filter Table (FHFT_FILTER). These contain separate values for each filter. Software must also enable the filter in the WUFC register, and enable the overall wake-up functionality must be enabled by setting *PME_En* in the PMCS register or the WUC register.

Structure of the Flexible Host Filter Table:

31	0	31	8	7	0	31	0	31	0
Reserved		Reserved		Mask [7:0]		DW 1		DW 0	
Reserved		Reserved		Mask [15:8]		DW 3		DW 2	
Reserved		Reserved		Mask [23:16]		DW 5		DW 4	
Reserved		Reserved		Mask [31:24]		DW 7		DW 6	

...

31	7	6	0	31	8	7	0	31	0	31	0
Reserved		Reserved		Reserved		Mask [127:120]		DW 29		DW 28	
Reserved		Length		Reserved		Mask [127:120]		DW 31		DW 30	

Each of the filters is allocated addresses as follows:

- Filter 0 — 0x09000 — 0x090FF
- Filter 1 — 0x09100 — 0x091FF
- Filter 2 — 0x09200 — 0x092FF
- Filter 3 — 0x09300 — 0x093FF
- Filter 4 — 0x09600 — 0x096FF
- Filter 5 — 0x09700 — 0x097FF
- Filter 6 — 0x09800 — 0x098FF
- Filter 7 — 0x09900 — 0x099FF

The following table lists the addresses used for filter 0.

Field	DWord	Address	Bit(s)	Initial Value
Filter 0 DW0	0	0x09000	31:0	X
Filter 0 DW1	1	0x09004	31:0	X
Filter 0 Mask[7:0]	2	0x09008	7:0	X
Reserved	3	0x0900C		X
Filter 0 DW2	4	0x09010	31:0	X
...				
Filter 0 DW30	60	0x090F0	31:0	X
Filter 0 DW31	61	0x090F4	31:0	X
Filter 0 Mask[127:120]	62	0x090F8	7:0	X
Length	63	0x090FC	6:0	X

Accessing the FHFT_FILTER registers during filter operation might result in a packet being mis-classified if the write operation collides with packet reception. It is therefore advised that the flex filters are disabled prior to changing their setup.

Once enabled, the flexible filters scan incoming packets for a match. If the filter encounters any byte in the packet where the mask bit is one and the byte does not match the byte programmed in the Flexible Host Filter Table (FHFT_FILTER), the filter fails that packet. If the filter reaches the required length without failing the packet, it passes the packet and generates a wake-up event. It ignores any mask bits set to one beyond the required length.

Packets that passed a wake-up flexible filter should cause a wake-up event only if it is directed to the E610 (passed L2 and VLAN filtering).

Note: The flexible filters are temporarily disabled when read from or written to by the host. Any packet received during a read or write operation is dropped. Filter operation resumes once the read or write access is done.

5.4.4 Wake-Up and Virtualization

When operating in a virtualized environment, all wake-up capabilities are managed by a single entity (such as the VMM or an IOVM). In an IOV architecture, the physical software device driver controls wake-up and none of the Virtual Machines (VMs) has direct access to the wake-up registers. The wake-up registers are not replicated.

5.5 Latency Tolerance Reporting (LTR)

The E610 generates PCIe LTR messages to report service latency requirements for snoop/non-snoop memory reads and writes to the root complex for system power management.

The E610 reports either latency tolerance or no latency tolerance requirements as a function of link, LAN port (link) and function status. The reported latency tolerance values are set to optimize platform power consumption without incurring packet loss due to receive buffer overflow.

5.5.1 LTR Algorithm

The E610 sends LTR messages following a functional change of the device. The E610 conglomerates the LTR latency requirements for its both functions as illustrated in [Figure 5-3](#) and detailed below:

1. The LTR functionality is enabled by the LTR capability structure of Function 0 in its PCIe configuration space.
2. When any function is at D0a with an active link and the last LTR message did not reported LTR requirements, The E610 initiates an LTR message with LTR requirements from the LTRC register.
3. When both functions are not at D0a with an active link and the last LTR message did not reported NO LTR requirements, The E610 initiates an LTR message with cleared LTR requirements flags.
4. When the OS clears the LTR enable flag in the LTR capability structure of Function 0, The E610 initiates an LTR message with cleared LTR requirements flags.
5. When the software device driver sets the `LTRC.LTR_SEND` bit in Function 0, The E610 initiates an LTR message with LTR requirements or no LTR requirements based on the `LTRC.LTR_REQ` flag.
6. The LTR requirements are collected from the LTRC register in Function 0.

5.5.2 LTR Initialization Flow

BIOS/OS configures the maximum snoop/non-snoop platform latency in the PCIe (function 0) configuration LTR capability registers.

- As part of software device driver initialization, the software device driver:
 - Reads the LTR capability to get the worst case latency tolerance requirement for the platform.
 - Auto negotiates extended LPI exit latencies.
 - The software device driver configures receive buffer watermarks for DMAC as aggressively as possible for the active target latency tolerance value (pre defined value).
 - The software device driver configures the content of the LTR snoop/non-snoop requirements to LTRC register in Function 0. Changing the LTR latency values in the LTRC register it is recommended to set also the `LTRC.LTR_SEND` flag with active `LTRC.LTR_REQ` to initiate an LTR message with these updated values.
- When a software device driver of function 0 unloads, it should clear the `LTRC.LTR_REQ` while setting the `LTRC.LTR_SEND` bit, causing an LTR message to be set. If function 1 is still active, the device will initiate additional LTR required message.

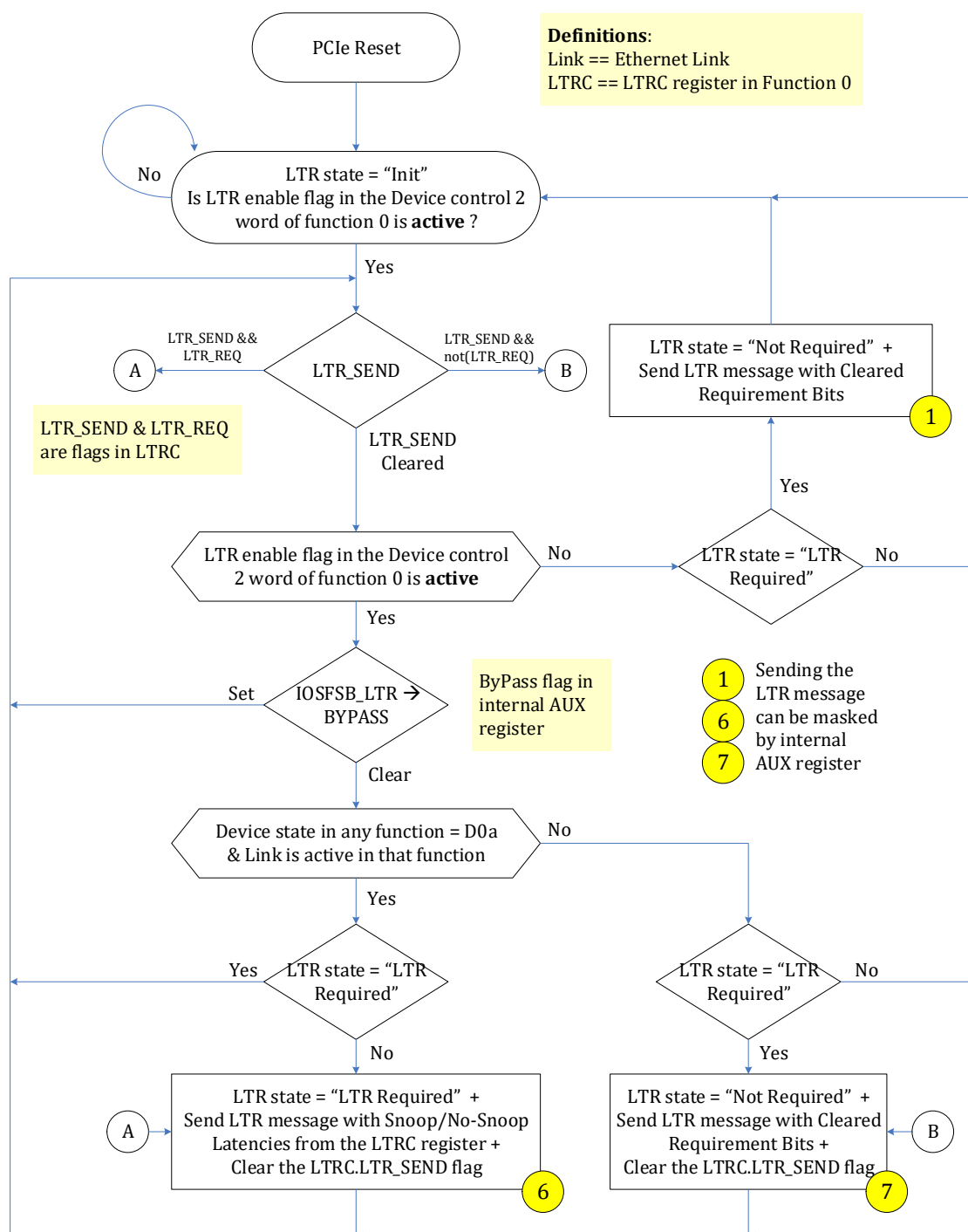


Figure 5-3. LTR Flow

NOTE: *This page intentionally left blank.*

Chapter 6 Non-Volatile Memory Map

6.1 NVM Organization

Note: All offsets in this chapter are relative to the E610 Configuration region. This region is located in the flash at offset 4 Kbyte as described in [Section 3.3.2.1](#). So to obtain the actual offset in the flash, 0x800 should be added to the requested word address.

6.1.1 NVM Map High Level

Table 6-1. Flat NVM Map - High Level

Area	Section	Size (words)	Description
Shadow RAM non-preserved	Init Module	256	pointers to other sections and legacy software word
Shadow RAM preserved	PFA Area	12 Kwords	A set of TLV sections as described in section TBD containing all the information that should be preserved across updates.
Shadow RAM - non preserved	Autoload sections	variable	Sections containing the autoloaded registers at different resets
	EMP configuration sections	variable	Sections containing configurations of the EMP. Includes EMP SR settings, manageability sections and OEM sections
Shadow RAM second copy	SRb	32 Kwords	A second copy of the Shadow RAM for updates.
Flash	NVM Bank 1		Signed section containing the basic map from which all the other maps are generated via adaptive NVM. It also contains a copy of the shadow RAM and extended Mini-loader
	NVM Bank 2		A second copy of the NVM Bank for updates.
	OROM Section - first copy		Signed section containing the option ROM image.
	OROM Section - second copy		A second copy of the OROM section for updates.
	Extended TLVs - first copy		
	Extended TLVs - second copy		
	Scratch pad area		
	Factory Settings		
	SPDM certificate slots		

6.1.2 NVM Header

Table 6-2 lists the format of the NVM header section. It includes words with a specific content and pointers to the first level of NVM modules. See Section 6.3 for more details.

- Items which are pointers to modules end with the pointer in there name.
 - The pointer to non shadow RAM modules which are part of NVM bank are relative to the beginning of the flash assuming 1st NVM bank is valid. In case the 2nd NVM bank is valid there is a need to add NVM Bank Area Size to the pointer.
 - The pointer to shadow RAM modules are relative to the beginning of the shadow RAM.
- The Auth / RO column indicates the following for modules pointed by pointers in the header:
 - Auth - The pointed module is authenticated
 - Adaptive - Format of pointed module is Authenticated, but content can be changed by adaptive NVM flows.
 - RW - The module is read/write

Note: A read-only item can be written only when either in the blank Flash programming mode or when NVM_SEC_EN bit is cleared in NVM word 0x02.

- The CSR Format column indicates whether the pointed module uses the formats described in Section 6.1.3.
- The Module is in Shadow RAM column indicates whether the module is mapped into the basic banks mirrored into the internal shadow RAM. If marked as No, the most significant bit of the pointer (bit15) must be set to 1b to indicate that the pointer value (in bits 14:0) is expressed in 4 KB sector units. Otherwise, it is expressed in word units.

Table 6-2. NVM Header Map

Word Address	Pointed Module Name / Word Name	For section [Auth]	Accessed by	Loading Trigger	CSR Format	Module is in Shadow RAM
0x00	NVM Control Word 1		EMP			
0x01	Non-persistent End Pointer		EMP			
0x02	Last PFA word Pointer					
0x03	GFID Pointer					
0x04	Reserved					
0x05	Reserved					
0x06	RO PCIR Registers Autoload Module Pointer					
0x07	Auto Generated Pointers Pointer	Auth	EMP/SW		No	Yes
0x08	Reserved					
0x09	EMP Global Module Pointer	Auth	EMP			
0xA	Guarded Zone Pointer					
0xB	EMP Image Pointer					
0xC	Reserved					
0xD	Reserved					
0xE	Manageability Module Pointer	Adaptive	EMP		No	Yes

Table 6-2. NVM Header Map [continued]

Word Address	Pointed Module Name / Word Name	For section [Auth]	Accessed by	Loading Trigger	CSR Format	Module is in Shadow RAM
0xF	EMP Settings Module Pointer					
0x10	SW Compatibility Word 1					
0x11	SW Compatibility Word 2					
0x12	SW Compatibility Word 3					
0x13	SW Compatibility Word 4					
0x14	SW Compatibility Word 5					
0x15	Reserved					
0x16	Reserved					
0x17	Reserved					
0x18	Software Reserved Word 1 - Dev Starter Revision					
0x19	SW Reserved Word 2		SW/BIOS	POR		
0x1A	SW Reserved Word 3		SW/BIOS	POR		
0x1B	SW Reserved Word 4		SW/BIOS	POR		
0x1C	SW Reserved Word 5		SW/BIOS	POR		
0x1D	SW Reserved Word 6		SW/BIOS	POR		
0x1E	SW Reserved Word 7		SW/BIOS	POR		
0x1F	SW Reserved Word 8		SW/BIOS	POR		
0x20	SW Reserved Word 9		SW/BIOS	POR		
0x21	SW Reserved Word 10		SW/BIOS	POR		
0x22	SW Reserved Word 11		SW/BIOS	POR		
0x23	SW Reserved Word 12		SW/BIOS	POR		
0x24	SW Reserved Word 13		SW/BIOS	POR		
0x25	SW Reserved Word 14		SW/BIOS	POR		
0x26	SW Reserved Word 15		SW/BIOS	POR		
0x27	SW Reserved Word 16		SW/BIOS	POR		
0x28	SW Reserved Word 17		SW/BIOS	POR		
0x29	SW Reserved Word 18 - Map version		SW/BIOS	POR		
0x2A	SW Reserved Word 19 - NVM_Image_Version		SW/BIOS	POR		
0x2B	SW Reserved Word 20 - NVM Structure Version		SW/BIOS	POR		
0x2C	SW Reserved Word 21		SW/BIOS	POR		
0x2D	SW Reserved Word 22 - EETRACK ID 1		SW/BIOS	POR		
0x2E	SW Reserved Word 23 - EETRACK ID 2		SW/BIOS	POR		
0x2F	Reserved					
0x30	Reserved					

Table 6-2. NVM Header Map [continued]

Word Address	Pointed Module Name / Word Name	For section [Auth]	Accessed by	Loading Trigger	CSR Format	Module is in Shadow RAM
0x31	Reserved					
0x32	Reserved					
0x33	IBA Capabilities		BIOS	POR		
0x34	SW Reserved Word 24 - Original EETRACK ID 1		BIOS	POR		
0x35	SW Reserved Word 25 - Original EETRACK ID 2		BIOS	POR		
0x36	Reserved					
0x37	Reserved					
0x38	Reserved					
0x39	Reserved					
0x3A	Reserved					
0x3B	Reserved (GLOBR)					
0x3C	Reserved (CORER)					
0x3D	PHY configuration scripts pointer (Dreadnought Lake)	Auth	EMP	POR	No	No
0x3E	Reserved					
0x3F	Reserved (was: Software Checksum) - moved to PFA					
0x40	Preserved Field Area pointer					
0x41	Reserved					
0x42	1st NVM Bank pointer					
0x43	NVM Bank Area Size					
0x44	1st OROM Bank pointer					
0x45	OROM Bank Area Size					
0x46	1st TLV extension bank pointer					
0x47	TLV Extension Bank Area Size					
0x48	EMP SR Settings Pointer					
0x49	Reserved					
0x4A	Reserved					
0x4B	Link Topology Scratch Pad Area Pointer		SW	POR		
0x4C	Link Topology Scratch Pad Area Size		SW	POR		
0x4D	Configuration Metadata Pointer	Auth	EMP		No	No
0x4E	Reserved					
0x4F	Reserved					
0x50	FW Scratch Pad Area Pointer	RW	EMP		No	No
0x51	FW Scratch Pad Area Size		EMP			

Table 6-2. NVM Header Map [continued]

Word Address	Pointed Module Name / Word Name	For section [Auth]	Accessed by	Loading Trigger	CSR Format	Module is in Shadow RAM
0x52	Reserved					
0x53	Reserved					
0x54	PHY subsystem post PLL configuration Module Pointer	Auth	EMP	POR	No	No
0x55	Soft SKUs		EMP		No	
0x56	Reserved					
0x57	Reserved					
0x58	Reserved					
0x59	Reserved					
0x5A	Reserved					
0x5B	Reserved					
0x5C	Reserved					
0x5D	White List Pointer		EMP			
0x5E	Sideband Autoload Pointer		EMP			
0x5F	RDE Dictionaries Pointer		EMP			
0x60	Reserved					
0x61	Factory Setting Pointer		EMP			
0x62	Factory Setting Size		EMP			
0x63	Mailbox register autoload pointer					
0x64	Reserved					
0x65	Reserved					
0x66	Signature ROT Key Revocation Module Pointer		EMP	POR	No	No
0x67	NVM ISK Module Pointer		EMP	POR	No	No
0x68	Reserved					
0x69	Reserved					
0x6A	Reserved					
0x6B	Reserved					
0x6C	SPDM Certificate slot 0 Pointer					
0x6D	SPDM Certificate slot 1 Pointer					
0x6E	SPDM Certificate slot 2 Pointer					
0x6F	SPDM Certificate slot 3 Pointer					
0x70	MAC Reset 0 Registers Autoload Pointer					
0x71	MAC Reset 1 Registers Autoload Pointer					
0x72	Master Reset 0 Registers Autoload Pointer					

Table 6-2. NVM Header Map [continued]

Word Address	Pointed Module Name / Word Name	For section [Auth]	Accessed by	Loading Trigger	CSR Format	Module is in Shadow RAM
0x73	Master Reset 1 Registers Autoload Pointer					
0x74	OEM Port Config 0 Pointer					
0x75	OEM Port Config Pointer					
0x76-0xFB	Spare NVM Header Words					
0xFC	Legacy NVM Control Word 1					
0xFD	Legacy NVM Control Word 2					
0xFE	Legacy NVM Control Word 3					
0xFF	Legacy NVM Control Word 4					

6.1.3 Structure of NVM Modules

An NVM module read by the EMP to configure hardware (as listed [Table 6-2](#)) is built according to one of the following structures:

- Fixed functionality structure — Each NVM word is allocated to one or more fixed fields and the contents of the NVM word are loaded to fixed CSRs in the E610. In [Table 6-2](#), these modules are referred as CSR format = No modules.
- Flexible functionality module — The structure defines the device address or addresses into which the NVM words are loaded. Therefore, the module might be used for different purposes based on its contents. In [Table 6-2](#), these modules are referred as CSR format = Yes modules.

The remainder of this section describes the structure of the contents of a flexible functionality module (excluding the module header described in [Section 6.1.5](#). The general structure of the module is shown in [Figure 6-1](#).

A CSR format module can be made of a mix of repeating Type 1, 2, 3 and 4 segments. Type 5 should be kept separate and can only be loaded by firmware without any hardware acceleration. Each segment is described by its own header (the shaded fields in [Figure 6-1](#)).

A sideband CSR format module includes a special Type F - subtype 0 segment, which includes the information needed to access the sideband module. The rest of the module can be made of a mix of repeating Type 1, 2, and 5 segments. This kind of module can only be loaded by firmware without any hardware acceleration <remove this statement if no hardware accelerator is used>. The Type F segment can repeat several times inside the module and is used to change access parameters to the sideband module. The sideband CSR format module is described in [Section 6.1.3.6](#).

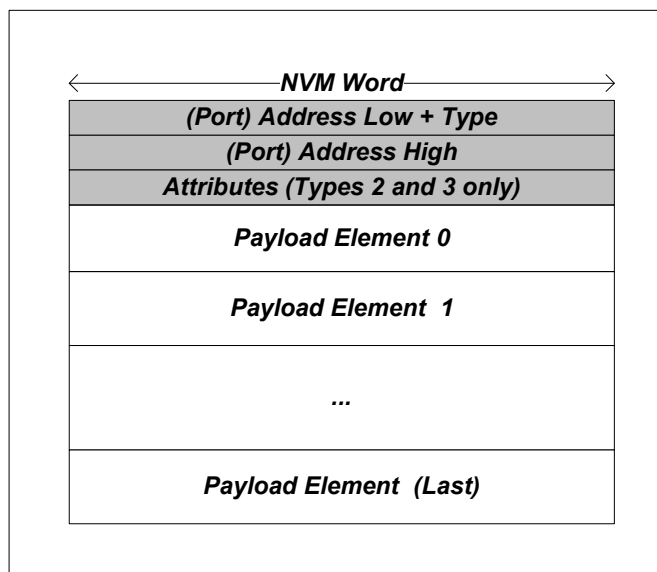


Figure 6-1. Structure of a Flexible Functionality Module

Description of the fields in [Figure 6-1](#):

- **(Port) Address** — A 28-bit value that defines the starting address to which data Dwords are loaded. Serves as either a direct address to load into (Types 1 & 2) or as an indirect address referred to as a port address (Type 3 & 4). A port is made of a 32-bit address register followed by data registers of total size width. Address low is a 12-bit field mapped to word[15:4].
- **Type** — A 4-bit command field mapped to word[3:0] bits that defines the type of flexible module. Descriptions are as follows:
 - Type 1 (0001b) — Loads a single 32-bit data segment
 - Type 2 (0010b) — Load a set of 32-bit data Dwords into consecutive addresses
 - Type 3 (0011b) — Loads a set of 32-bit data Dwords through an address port
 - Type 4 (0100b) — Loads, through an address port, a sequence of data blocks into arbitrary addresses
 - Type 5 (0101b) - Loads a single 64-bit data segment
 - Type F (1111b) - Used to define access parameters to sideband module.
- **Attributes** — A 16-bit optional word that defines attributes of the module
 - **Width** — A 3-bit field mapped to word[2:0] bits that describes the width (in 32-bit Dwords) of a data element
 - 000b = Data is 32-bit wide
 - 001b = Data is 64-bit wide
 - 010b = Data is 128-bit wide
 - 011b = Data is 256-bit wide
 - Else = Reserved
 - **Skip** — A 2-bit field mapped to word[4:3] bits that describes the number of address bytes to be skipped for getting the address of the next payload element (Type 2) or port register involved (Type 3 or 4).
 - 00b = Skip 4 bytes
 - 01b = Skip 8 x 4 bytes = 32 bytes
 - 10b = Skip 32 x 4 bytes = 128 bytes
 - 11b = Skip 0 bytes
 - **Length** — An 11-bit field mapped to word[15:5] bits that contains the number of data elements, each of Width bits. length of zero is illegal.
- **Payload Elements** — A sequence of address or data elements to be loaded into the E610. The structure of the Payload field varies with each type and is described in the sections that follow.

6.1.3.1 Type 1 Module

Used to specify a single 32-bit data segment. The *Address* field defines the address into which the 32-bit data is loaded. See [Figure 6-2](#).

The common use of a Type 1 module is to concatenate a list of such structure to load a set of 32-bit values into various CSRs.

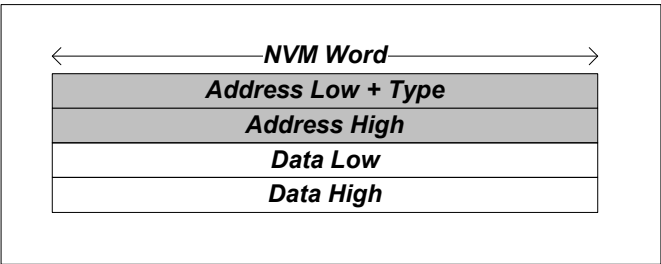


Figure 6-2. Structure of a Type 1 Module

6.1.3.2 Type 2 Module

Used to load a sequence of data into consecutive addresses, such as a memory array. See [Figure 6-3](#).

The *Address* field defines the first address to be loaded. Data is loaded as width-size elements into consecutive addresses. The *Attributes* field applies the following information:

- Width = 000b
- Skip = see previous description
- Length = see previous description

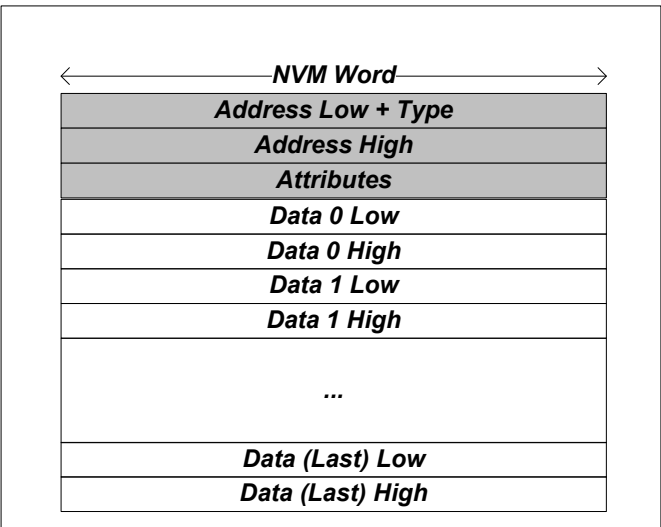


Figure 6-3. Structure of a Type 2 Module

6.1.3.3 Type 3 Module

Used to load a sequence of data through an address port.

Figure 6-4 shows the structure of a Type 3 module. The following fields have special values:

- The *Port Address* field defines the address of the port address register through which address/data is loaded. The Port Address register is followed by the Port Data registers.
- The *Attributes* field applies with the following fields:
 - Width — The size of a single data element written into the port during one port load cycle
 - Length — Number of data elements written through the port
 - Skip — Encodes the number of bytes between the addresses of the port registers involved
- The *Indirect Address* field defines the first consecutive address to write to. It is written into the Port Address register for the first item and from then on the indirect address is increased (by E610) with each write of a new data element from the list. Consecutive data elements are written into the port.
- The *Data Elements* field are made of length elements, each of the width DWords. Each port load cycle handles one data element

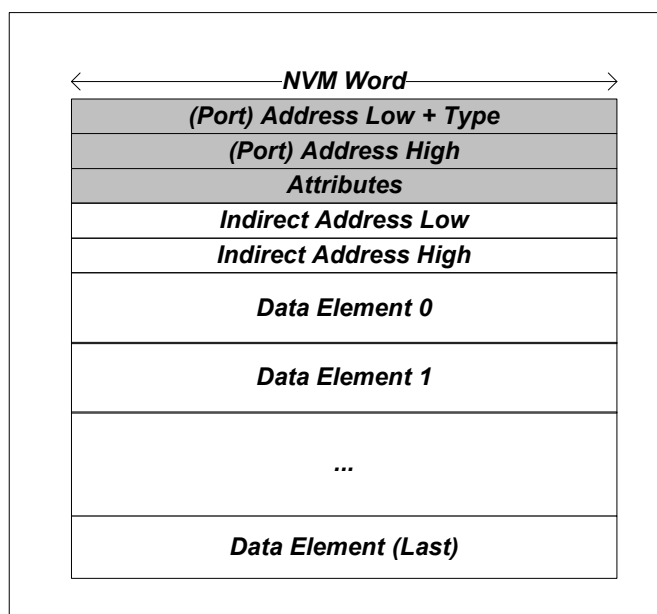


Figure 6-4. Structure of a Type 3 Module

The following figures describe two cases of using a Type 3 module:

- Figure 6-5 shows a case of loading 32-bit values into a Global (GL) port.
- Figure 6-6 shows a case of loading 64-bit values into a PRT port.

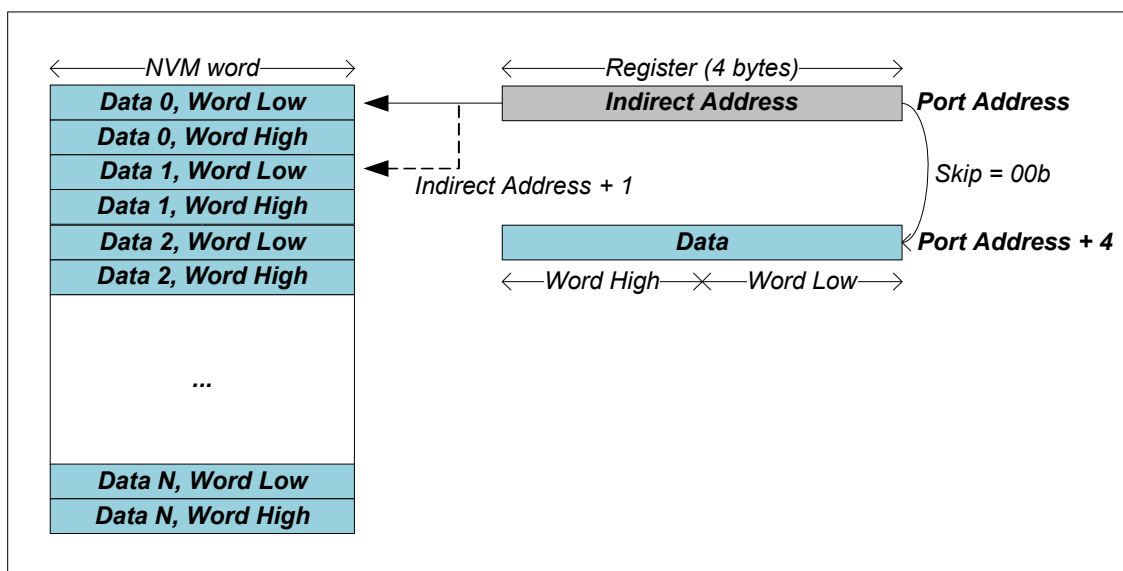


Figure 6-5. Access Through a Type 3 Module of 32-Bit Wide Data Elements

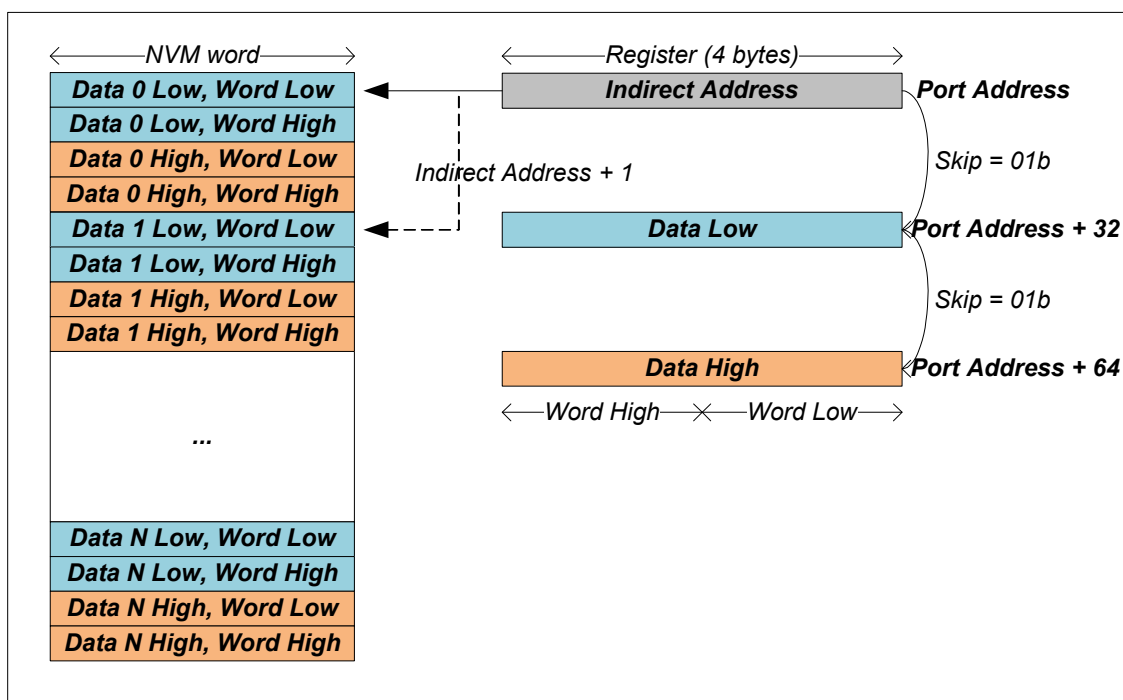


Figure 6-6. Access Through a Type 3 Module of 64-Bit Wide per-Port Data Elements

6.1.3.4 Type 4 Module

Used to load, through an address port, a sequence of scattered data elements at arbitrary addresses.

Figure 6-7 shows the structure of a Type 4 module. The following fields have special values:

- The *Port Address* field defines the address of the port address register through which address/data is loaded. The port address register is followed by the port data registers.
- The *Attributes* field applies with the following fields:
 - Width — The size of a single data element written into the port during one port load cycle
 - Length — Number of data elements written through the port, not including the indirect address words (the words colored in gray in Figure 6-8 and Figure 6-9).
 - Skip — Encodes the number of bytes between the addresses of the port registers involved
- The *Indirect Address* fields define the address that the following data element is written into. It is written into the Port Address register.
- The Data Element is written into the port data register. The sequence of writing (the indirect address and its data element) into the port is repeated per each data element (length times).

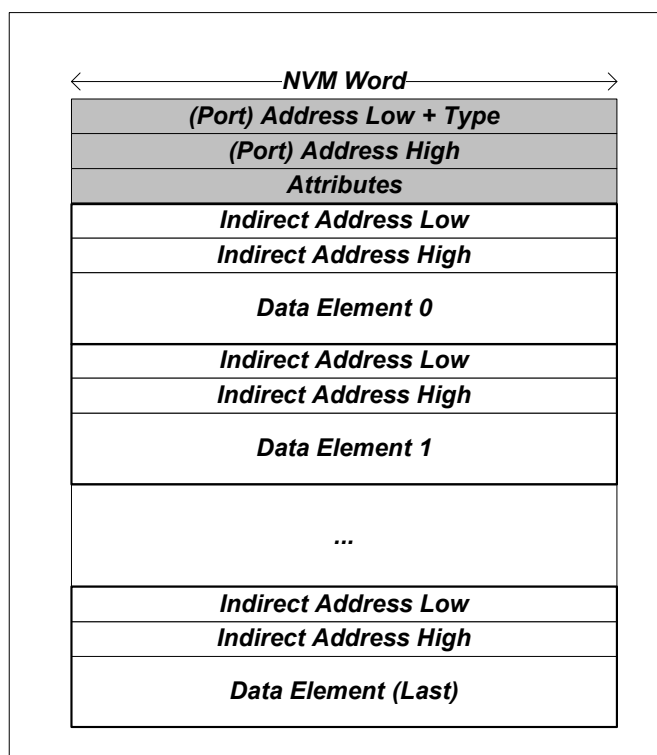


Figure 6-7. Structure of a Type 4 Module

The following figures show two cases of using a Type 4 module:

- Figure 6-8 shows a case of loading 32-bit values into a global (GL) port.
- Figure 6-9 shows a case of loading 64-bit values into a PRT port.

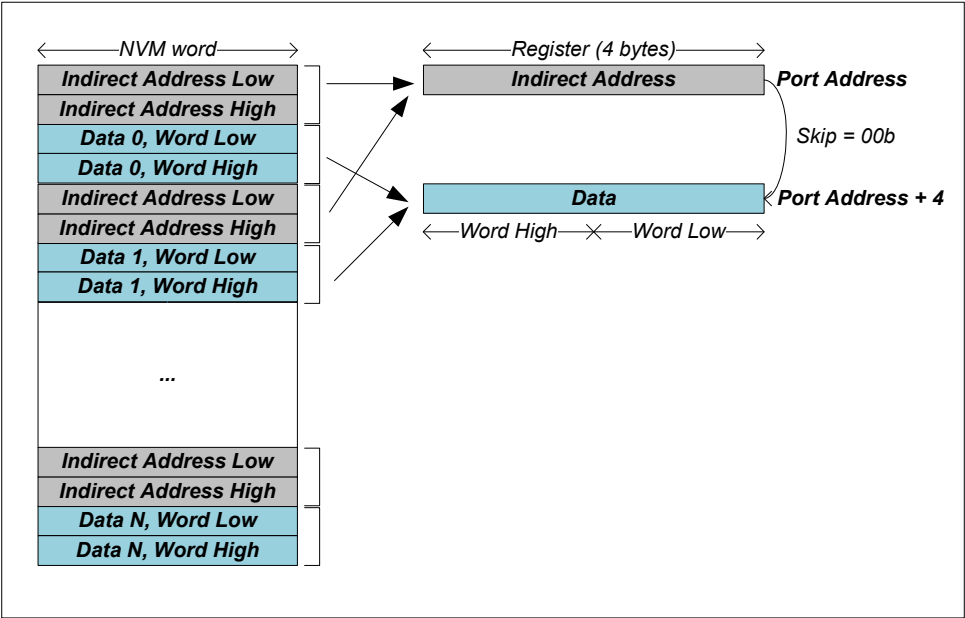


Figure 6-8. Access Through a Type 4 Module of 32-Bit Wide Data Elements

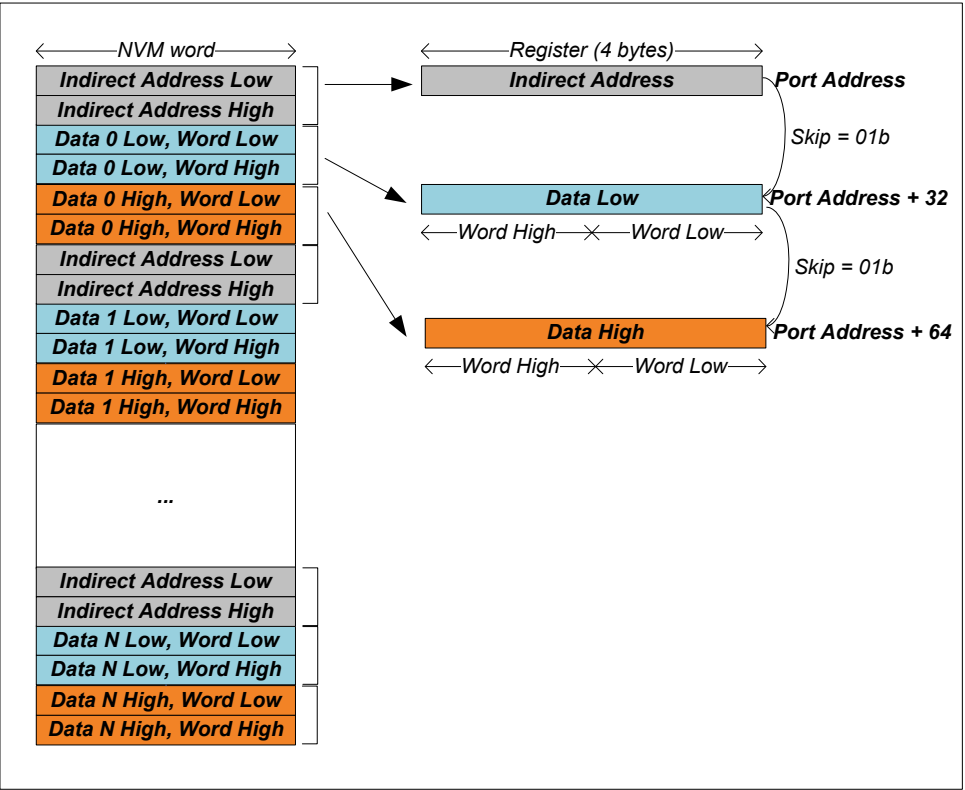


Figure 6-9. Access Through a Type 4 Module of 64-Bit Wide per-Port Data Elements

6.1.3.5 Type 5 Module

Used to load, a single 64-bit data segment addressed by 32-bit address. Firmware handles this module type without any hardware acceleration. It is intended for loading into external SB-IOSF IPs.

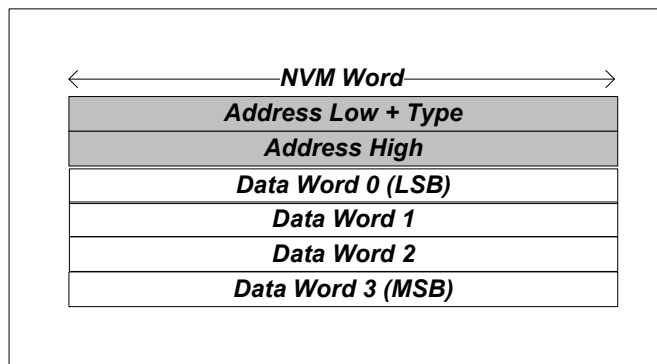


Figure 6-10. Structure of a Type 5 Module

6.1.3.6 Type F Module

This is a special type, which is not used to directly load a specific address, but it is used to indicate specific access conditions for the register in the autoload module. Specifically, subtype 0 within type F module indicates that this module is used to load sideband device. Type F module should be followed by one or more Type 1,2 or 5 modules, which indicates the actual data to load. Figure 6-11 describes the format of Type F, subtype 0.

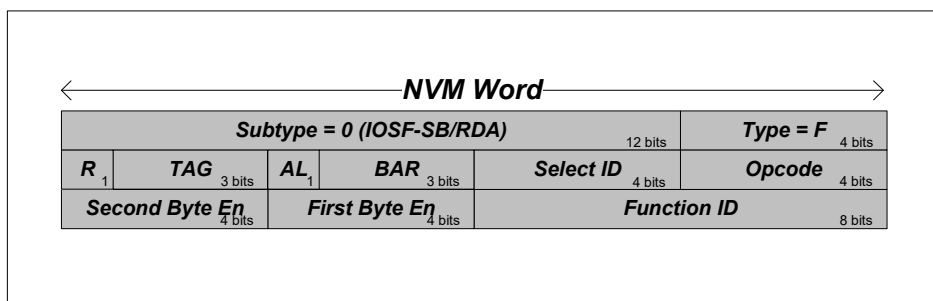


Figure 6-11. Structure of a Type F Module

Two additional 16 bits words are attached to Type F, subtype 0, with the following indications:

- **Opcode** — A 4-bit value that defines the sideband opcode that is used for loading the data in the following type 1,2 or 5 modules. The opcode should be selected from Table 3-39. Type 1 and 5 should typically use post or non-posted MWr or CRWr opcodes. Type 2 might use posted or non-posted MBWr opcode.
- **Select ID** — Indicates the sideband device to be loaded. The device should be selected from Table 3-38.
- **BAR** — Indicates BAR number of the sideband address (ignored - set to zero).

- **AL** — Address length flag. Indicates the length of the sideband address. For a 16 bits address, the 16 low significant bits are taken from the 28 bits of Type 1,2 or 5 address. For a 48 bits address, the 28 bits of Type 1,2 or 5 address are taken and padded with 20 bits of 0.
 - 0 - 16 bits address
 - 1 - 48 bits address
- **TAG** — Indicates TAG field of the sideband command (ignored - set to zero).
- **R** — Reserved, Set to 0.
- **Function ID** — Indicates the function ID of the sideband transaction (ignored - set to zero).
- **First Byte Enable** — Indicates the byte enable bits for the first 32 bits word written.
- **Second Byte Enable** — Indicates the byte enable bits for the second 32 bits word written.

Figure 6-12 illustrates the format of a Type F Autoload section.

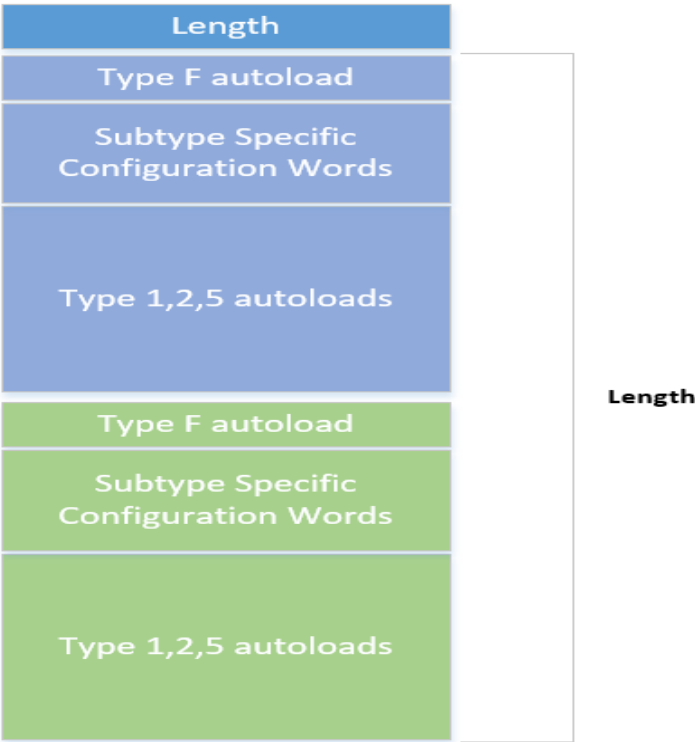


Figure 6-12. Type F Autoload Section

6.1.3.7 Format of the Hardware Modules in Flash

Some modules are not loaded directly from the shadow RAM but read from flash to the local RAM by firmware before loading to the hardware. Such modules could be too big to be copied to the local RAM as a single chunk. They could also be too big to be loaded with the hardware accelerator in a single run.

The module structure described shown in Figure 6-13 allows reading the module from NVM in chunks, which fit the allocated buffer in the local RAM and can be sent to the hardware accelerator. The chunk size is up to 4KBytes. All length values are in 16-bit words not including the length itself. The firmware reads the module content chunk by chunk to the local RAM and runs the hardware accelerator for the loaded chunk.

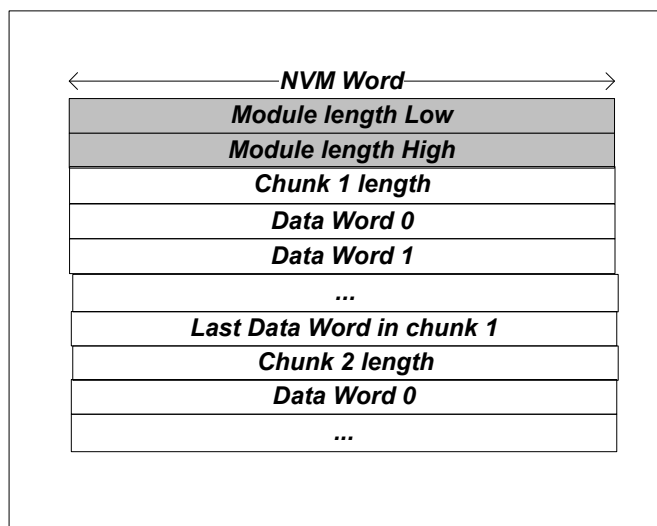


Figure 6-13. Format of Hardware Modules in Flash

6.1.3.8 List of CSRs Excluded

Some CSRs may contain fields that are critical for the host to device connectivity or for the NVM authentication scheme. EMP shall abstain from loading these CSRs via one of the 4 types listed above. If one of the following CSRs is included in a CSR format NVM module, hardware autoload process skips it and move forward to load the next CSR.

- List of protected registers (by address ordering) as defined by GLNVM_PROTCSR.

Table 6-3. Protected CSR List

Protected CSR	CSR address	NVM Protected (defined by GLNVM_PROTCSR array)
SB_IOSF_CTRL2	0x0A4000	Yes
SB_IOSF_CTRL1	0x0A4004	Yes
GLGEN_STAT	0x0B612C	Yes
GLNVM_ALTIMERS	0x0B6140	Yes
GL_MNG_FLASH_HIDDEN_AREA	0x0B7008	Yes

Defaults to these protected CSRs can be configured from a RO module that is referred as the CSR Protected List module (pointed by NVM word 0x0D). Any attempt to load one of these registers from another NVM module is dropped by the device. The CSR Protected List module is made of two Type 1/2 sections which shall appear in the following order:

1. Default settings to the protected CSRs
2. Contents of the GLNVM_PROTCSR array of registers. These registers shall be set with the addresses of the protected CSRs.

The CSR Protected List module is loaded only at POR and before any other module which is loaded into the shadow RAM. The contents of RO modules is not checked against the CSR Protected List contents.

The CSR Protected List module shall include only items which are either used in the modules listed above, or which are autoloading from NVM on POR events only, or which are loaded by EMP.

6.1.3.9 Auto-Generated Pointers

Type 1 and Type 2 items are generated by the auto map generation tool from the project's register database. These items are automatically mapped into the Registers Autoload NVM modules according to their loading trigger (see [Table 6-2](#)) and with no possibility to perform manual changes. Until the list of Type 1/2 items stabilizes, each invocation of the tool might lead to a different mapping offset in the destination module.

EMP code and software tools cannot accommodate these variations in mapping offset, and hence, they need a fixed method for accessing some predefined Type 1/2 items. The auto-generated pointers module is automatically defined by the auto map generation tool for this purpose. It is pointed to by NVM word 0x07. It lists the mapping address in NVM of the pre-defined Type 1/2 items via a 2-word structure per item.

The word address in shadow RAM of an item is given by the sum of the contents of its two associated words: pointer + offset. If the item is relative to an array of registers and/or to a register with a scope different than global, the offset is given for the first data word of the Type 1/2 array, for array instance [0] and scope instance [0].

In the auto-generated pointers module, the location of the 2-word structures relative to an item is made invariant along the project's life. The alias name of the Type 1/2 register appears in the names of the two words in the structure.

See Auto Generate Pointers Module on the detailed NVM map for the list of auto-generated pointers ([Section 6.3.52](#)).

6.1.4 NVM Integrity Checks by Software

This section describes the NVM integrity fields inserted in the E610 NVM map to provide a basic detection of a faulty Flash part.

A software checksum check is performed by the PF driver just after completion of its initialization sequence. It covers only PFA modules and the init section modules which are mapped into shadow RAM. If the checks fails, another try is attempted, and if it fails again, the PF driver disables Tx/Rx operations with E610.

Modules mapped outside the shadow RAM are signed, except for the scratch pad areas. EMP is responsible to check the signature of the modules at POR/EMPR.

6.1.4.1 Software Checksum

The Software Checksum section (TLV type 0x3F in PFA) covers the PFA content (reserved words included).

The Software Checksum word in the TLV above is computed such that after adding all the covered words, including the Software Checksum word itself, the sum is 0xBABA.

Each time EMP is updating the contents of one of these areas by its own initiative (not as part of an NVM Update AQ command) or for handling an MC command received over SMBus or NC-SI, it also updates the Software Checksum field accordingly.

The software device driver can check if the checksum is valid using the NVM Checksum Command (0x0706). See Section [Section 3.3.10.7](#).

6.1.5 Header of NVM Modules

6.1.5.1 Header of All NVM Modules Mapped to Shadow RAM

Table 6-4. NVM Header of Modules Mapped to Shadow RAM

Number of words	Field or Segment Name	Description and Comments
1	Module Length	Length of the module contents expressed in words (Module Length field excluded). It must be set to N. Modules are size limited to the size of the shadow RAM (64 KB).
N	Word 1	
	Word 2	
	...	
	Word N	

6.1.5.2 Header of Authenticated NVM Modules

This section concerns the following modules:

1. NVM Bank (pointed by NVM word 0x42)
2. Option ROM (pointed by NVM word 0x44)
3. Netlist (pointed by NVM word 0x46)
4. Extended Mini-loader
5. Recovery Firmware
6. ISK Modules
7. Key Revocation Modules

[Table 6-5](#) shows the header of the Firmware Image module. Fields colored in light blue are protected by the authentication signature.

Table 6-5. Header of Authenticated Modules

Number of Words	Field or Segment Name	Description and Comments
64	Header	
192	RSA Public Key	This field is skipped for the SHA384 Hash computing.
192	RSA Public Key Prime	Montgomery Constant n' - Precomputed value used for RSA 3K signature validation
192	RSA Public Key Second	Montgomery Constant n'' - Precomputed value used for RSA 3K signature validation
2	RSA Exponent	This field is skipped for the sake of SHA256 Hash computing.
192	Encrypted SHA Hash	This field is skipped for the sake of SHA256 Hash computing.
1	E610 Blank NVM Device ID	A unique ND-provided device ID that identifies the E610 controller among other ND controllers. It must be set to 0x57AC in the E610.
2	Max Module Area	It is the maximum flash area expressed in words that can be used by the module, starting from authentication header (included).
2	Current Module Area	It is the flash area expressed in words that is currently used by the module, starting from authentication header (included).
1	Reserved	Reserved
1	Code Revision	Bits[15:8] = Major revision number. Bits[7:0] = Minor revision number.
1	Reserved Spare Word	Must be zeroed.
N	word 1	Content of the authenticated module
	word 2	
	...	
	word N	

6.1.5.3 Module TypeIDs

Each module has a TypeID. The TypeID uniquely identify the module. TypeID is used in admin commands to access the module. All modules can be read through the TypeID with out knowing the exact location of the module in the NVM. PFA modules which allow direct update can be written using their TypeID. The table below list the TypeIDs of all of the modules in the NVM.

Table 6-6 list the TypeIDs of all of the modules in the NVM.

Table 6-6. Module TypeID Table

Module TypeID	Section	in PFA	Read Only
0x00	Reserved (Has special meaning in admin commands)		N/A
0x03	GFID Module		
0x06	Reserved		
0x07	Auto Generated Pointers Module		RO
0x08	Reserved		
0x09	EMP Global Module		RO
0x0A	Guarded Zone Module		RO

Table 6-6. Module TypeID Table [continued]

Module TypeID	Section	in PFA	Read Only
0x0B	EMP Image Module		RO
0x0C	Reserved		
0x0E	Manageability Module		RO
0x0F	EMP Settings Module		RO
0x16	PBA Block Module		RO after manufacturing
0x2F	VPD Module	Yes	RW via VPD access
0x30	PXE Setup Options	Yes	
0x31	PXE Configuration Customization Options	Yes	
0x32	PXE Version	Yes	
0x33	IBA Capabilities	Yes	
0x37	VLAN Configuration Block		
0x38	Reserved		
0x3B	Reserved		
0x3C	Reserved		
0x3D	PHY configuration scripts (Dread-naught Lake) Module		RO
0x3F	PFA checksum module	Yes	RO
0x40	Preserved Field Area Module	Yes	RO
0x42	1st NVM Bank Module		Writable through authenticated update
0x44	1st OROM Bank Module		Writable through authenticated update
0x46	1st Topology Netlist bank Module		
0x48	EMP SR Settings Module		RO
0x49	Feature Configuration Module	Yes	Writable via ANVM only
0x4A	Reserved		
0x4B	Link Topology Scratch Pad Area Module		RO
0x4D	Configuration Metadata Module	No	RO
0x4E	Immediate Values Module	Yes	Writable via ANVM only
0x4F	Reserved		
0x50	FW Scratch Pad Area Module		RW
0x53	Reserved		
0x54	PHY subsystem post PLL configuration Module	No	RO
0x56	Reserved		
0x57	Recovery FW	No	
0x58	Reserved		
0x5D	Allow List	No	
0x5E	Sideband Autoload	No	
0x5F	RDE Dictionaries	No	
0x61	Factory Settings	No	

Table 6-6. Module TypeID Table [continued]

Module TypeID	Section	in PFA	Read Only
0x107	MNG Filters Module	Yes	RW
0x108	Pass-through Control Words Structure 0 Module	Yes	RW
0x109	Pass-through Control Words Structure 1 Module	Yes	RW
0x10A	Reserved		
0x10B	Reserved		
0x10C	Original EETrack ID Module	Yes	
0x10F	PF MAC Address Module	Yes	RW
0x110	MNG MAC Address Module	Yes	RW
0x113	Early PCIR autoloading Module (512 bytes)	Yes	RO
0x114	Reserved		
0x115	Reserved		
0x116	Reserved		
0x117	Reserved		
0x118	Early POR autoloading module (512 bytes)	Yes	RO
0x119	Reserved		
0x11B	Link Topology Netlist Module	No	Updatable via Netlist update flow
0x11D	UART Debug Defaults Module	Yes	RW
0x120	FW MNG Link Topology Module	Yes	RW
0x127	PLDM Component Image Set Version String	Yes	RW
0x128	OEM Specific Module	Yes	RW
0x130	MinSrev	Yes	RW (with limitations)
0x131	RDE Parameters	Yes	RW
0x132	Combo Image Version Descriptor (CIVD)	Yes	RW (as part of NVM update)
0x133	PCIe Serial Number	Yes	RO after manufacturing
0x134	Link Default override	Yes	RW
0x136	RDE Ethernet MTU	Yes	RW
0x139	HII port disable by function	Yes	RW
0x146	Netlist MinSrev	Yes	RW (with limitations)
0x14A	OEM RoT Key Revocation Module	Yes	RW
0x14B-0xFFE	Reserved		
0xFFFF	Padding Module	Yes	RO

6.1.5.4 Preserved Fields Area structure

The PFA is a structure built as a set of TLVs as follow:

- PFA Header - First word of PFA is the Length.
- PFA Sub-Modules Format -

The PFA Sub-Modules is in Type-Length-Data (TVL) Format.

Every TLV type appears only once

TLVs can come in any order, in Shadow RAM or in the Extension TLV area.

TLVs Format:

- TypeID – 1 Word (taken from Table 6-6 only types which exists in PFA)
- Length – 1 Word: Max area provisioned for the section.
- Data – a variable number of words according to the “length” field.

6.1.6 Adaptive NVM Structures

6.1.6.1 Metadata Structure

Metadata is a structure within the signed area that describes the different features, immediate fields, and information on how to modify them.

The structure of the metadata is as follows:

Section Length (1W)	Super Feature Module Offset (1W)	Immediate Module Offset (1W)	Reserved (1W)
Features Fields Module			
Super-Features Fields Module			
Immediate Fields Module			
Description Text Module			
CRC8 (1W)			

6.1.6.1.1 Feature Fields Module

The feature fields module contains the following sub-modules:

- Features Header Array
 - Contain a list of features, where each feature specifies number of options and fields.
 - Points to feature configuration pointer array and feature field description.
- Features Configuration Option Array
 - Contain a list of options, where each option points to a feature data array.
- Features Field Descriptor Array
 - Contain a list of fields, where each field is identified by its NVM address and mask.

- Features Data Array
 - Contains a list of data per field.

Figure 6-14 describes the connection between the various feature tables.

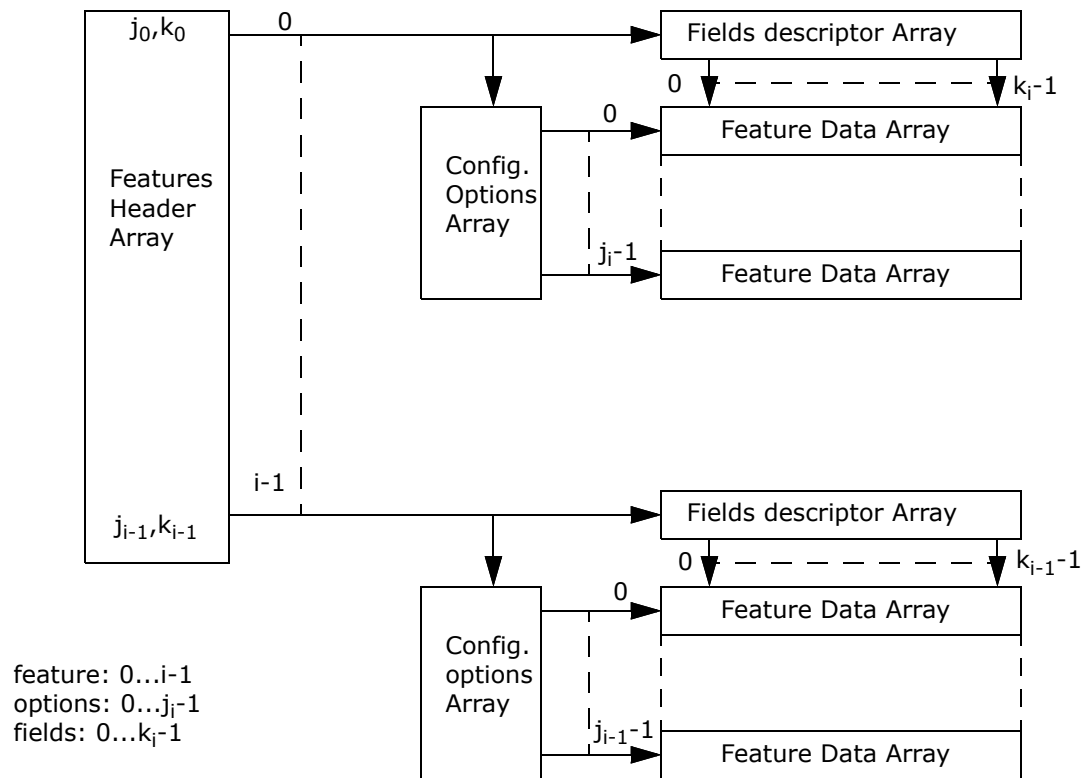


Figure 6-14. Feature Tables Diagram

6.1.6.1.1.1 Features Header Array

The features header array is built as follows:

Number of Features (1W)						
Feature ID (0) 1W	Pointer ¹ to Description Text 1W	Feature (0) Flags 1W	Number of Options 1W	Number of Fields 1W	Pointer to Options Array 1W	Pointer to Fields Descriptor Array 1W
Feature ID (1) 1W	Pointer to Description Text 1W	Feature (1) Flags 1W	Number of Options 1W	Number of Fields 1W	Pointer to Options Array 1W	Pointer to Fields Descriptor Array 1W
...						
Feature ID (n) 1W	Pointer to Description Text 1W	Feature (n) Flags 1W	Number of Options 1W	Number of Fields 1W	Pointer to Options Array 1W	Pointer to Fields Descriptor Array 1W

1. The pointer is the word offset inside the metadata.

The feature flags are as follows:

Bit(s)	Content
15:4	Reserved
3	DWord
2	Hidden ¹
1	Reserved
0	OEM ²

1. The hidden flag is set by default for features included in a super-feature.

2. Feature cannot be set by software as adaptive NVM feature. It is set during map generation per OEM.

6.1.6.1.1.2 Features Configuration Option Array

The features configuration option array is described as follows:

Option ID (0) 1W	Pointer ¹ to Option Description Text 1W	Pointer in Features Data Array 1W
Option ID (1) 1W	Pointer to Option Description Text 1W	Pointer in Features Data Array 1W
...	...	...
Option ID (n) 1W	Pointer to Option Description Text 1W	Pointer in Features Data Array 1W

1. The pointer is the word offset inside the metadata.

6.1.6.1.1.3 Feature Field Descriptor Array – Case 1

- DWord flag = 0
- i = 0...Number of features -1
- j = 0...Number of fields -1

Field[0] (0) Address ¹ 2W	Field[0] (0) Mask 1W	...	Field[0] (j) Address 2W	Field[0] (j) Mask 1W
Field[1] (0) Address 2W	Field[1] (0) Mask 1W	...	Field[1] (j) Address 2W	Field[1] (j) Mask 1W
Field[i] (0) Address 2W	Field[i] (0) Mask 1W	...	Field[i] (j) Address 2W	Field[i] (j) Mask 1W

1. The address is the absolute word address from the beginning of the NVM.

6.1.6.1.1.4 Feature Field Descriptor Array – Case 2 (Generic)

- DWord flag = 1
- i = 0...Number of features -1
- j = 0...Number of fields -1

Field[0] (0) Address ¹ 2W	Field[0] (0) Mask LSB 1W	Field[0] (0) Mask MSB 1W	Reserved 2W	...	Field[0] (j) Address 2W	Field[0] (j) Mask LSB 1W	Field[0] (j) Mask MSB 1W	Reserved 2W
Field[1] (0) Address 2W	Field[1] (0) Mask LSB 1W	Field[1] (0) Mask MSB 1W	Reserved 2W	...	Field[1] (j) Address 2W	Field[1] (j) Mask LSB 1W	Field[1] (j) Mask MSB 1W	Reserved 2W
Field[i] (0) Address 2W	Field[i] (0) Mask LSB 1W	Field[i] (0) Mask MSB 1W	Reserved 2W	...	Field[i] (j) Address 2W	Field[i] (j) Mask LSB 1W	Field[i] (j) Mask MSB 1W	Reserved 2W

1. The address is the absolute word address from the beginning of the NVM.

6.1.6.1.1.5 Feature Field Descriptor Array – Case 2

- DWord flag = 1
 - If field is a POR CSR:
 - THEN the CSR address is attached.
 - ELSE 0xFFFFFFFF to other fields of this feature.
- i = 0...Number of features -1
- j = 0...Number of fields -1

Field[0] (0) Address ¹ 2W	Field[0] (0) Mask LSB 1W	Field[0] (0) Mask MSB 1W	Field[0] (0) CSR Address ² LSB 1W	Field[0] (0) CSR Address MSB 1W	...	Field[0] (j) Address 2W	Field[0] (j) Mask LSB 1W	Field[0] (j) Mask MSB 1W	Field[0] (j) CSR Address LSB 1W	Field[0] (j) CSR Address MSB 1W
Field[1] (0) Address 2W	Field[1] (0) Mask LSB 1W	Field[1] (0) Mask MSB 1W	Field[1] (0) CSR Address LSB 1W	Field[1] (0) CSR Address MSB 1W	...	Field[1] (j) Address 2W	Field[1] (j) Mask LSB 1W	Field[1] (j) Mask MSB 1W	Field[1] (j) CSR Address LSB 1W	Field[1] (j) CSR Address MSB 1W
Field[i] (0) Address 2W	Field[i] (0) Mask LSB 1W	Field[i] (0) Mask MSB 1W	Field[i] (0) CSR Address LSB 1W	Field[i] (0) CSR Address MSB 1W	...	Field[i] (j) Address 2W	Field[i] (j) Mask LSB 1W	Field[i] (j) Mask MSB 1W	Field[i] (j) CSR Address LSB 1W	Field[i] (j) CSR Address MSB 1W

1. The address is the absolute word address from the beginning of the NVM.
2. The CSR address is added to enable E610 firmware patches (update CSR on next reset). The CSR address is the physical address of this CSR.

6.1.6.1.1.6 Feature Data Array

The fourth part of a featured fields module includes a 3D (ixjxk) feature data array.

Each data cell is by default 1W width.

If a DWord is set, it is 2W.

- i = 0...Number of features -1
- j = 0...Number of options -1
- k = 0...Number of fields -1

Note: Number of options and Number of fields are feature-dependent. See [Section 6.1.6.1.1.1](#).

Feature(0)			
Data[0] (0)	Data[0] (1)	...	Data[0] (k)
Data[1] (0)	Data[1] (1)	...	Data[1] (k)
Data[j] (0)	Data[j] (1)	...	Data[j] (k)
Feature(1)			
Data[0] (0)	Data[0] (1)	...	Data[0] (k)
Data[1] (0)	Data[1] (1)	...	Data[1] (k)
Data[j] (0)	Data[j] (1)	...	Data[j] (k)
Feature(i)			
Data[0] (0)	Data[0] (1)	...	Data[0] (k)
Data[1] (0)	Data[1] (1)	...	Data[1] (k)
Data[j] (0)	Data[j] (1)	...	Data[j] (k)

6.1.6.1.2 Super-Feature Fields Module Structure

The super-feature fields module contains the following sub-modules:

- Super-Features Header Array
 - Contains a list of supper-features, where for each super-feature, the number of options and number of Feature Configuration IDs (FCIDs) is specified.
 - Points to Super-features Configuration Option Array.
- Super-Features Configuration Option Array
 - Contain a list of options, where each options points to a Super-Features Data Array.
- Super-Features Data Array
 - Contain a list of (Feature ID, Feature Configuration option number) pairs.

Figure 6-15 describes the connection between the various feature tables.

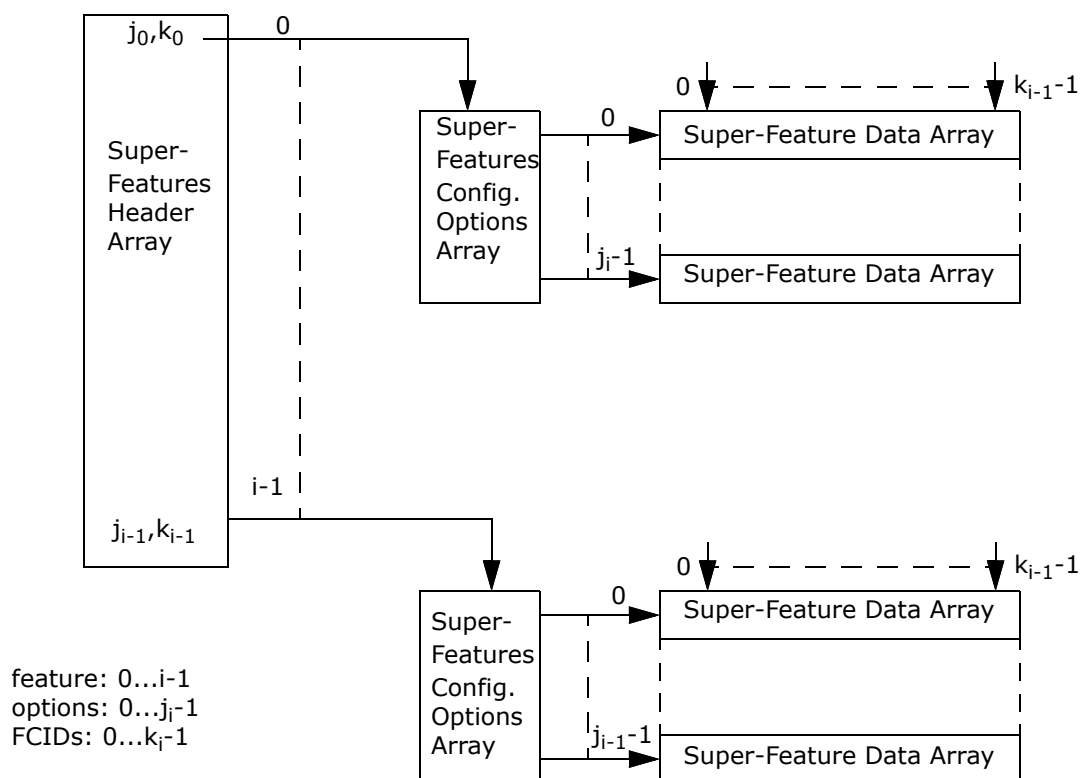


Figure 6-15. Super-Feature Tables Diagram

6.1.6.1.2.1 Super-Feature Header Array

Number of Super-Features (1W)					
Super-feature ID (0) 1W	Pointer ¹ to Description Text 1W	Feature (0) Flags ² 1W	Number of Options 1W	Number of selectable FCIDs 1W	Pointer to Options Array 1W
Super-feature ID (1) 1W	Pointer to Description Text 1W	Feature (1) Flags 1W	Number of Options 1W	Number of Selectable FCIDs 1W	Pointer to Options Array 1W
...					
Super-feature ID (n) 1W	Pointer to Description Text 1W	Feature (n) Flags 1W	Number of Options 1W	Number of Selectable FCIDs 1W	Pointer to Options Array 1W

1. The pointer is the word offset inside the metadata.
2. Feature flags bit 0 is the OEM flag. Other bits are reserved in this DCR.

6.1.6.1.2.2 Super-Feature Configuration Option Array

Super-feature Option ID (0) 1W	Pointer ¹ to Option Description Text 1W	Pointer in Super-Features Data Array 1W
Super-feature Option ID (1) 1W	Pointer to Option Description Text 1W	Pointer in Super-Features Data Array 1W
...	...	...
Super-feature Option ID (n) 1W	Pointer to Option Description Text 1W	Pointer in Super-Features Data Array 1W

1. The pointer is the word offset inside the metadata.

6.1.6.1.2.3 Super-Feature Data Array

3D (ix)jk super-feature data array with a 2W data size.

- i = 0...Number of super-features -1
- j = 0...Number of options -1
- k = 0...Number of FCIDs -1

Super-Feature(0)			
Data[0] (0) 2W	Data[0] (1) 2W	...	Data[0] (k) 2W
Data[1] (0) 2W	Data[1] (1) 2W	...	Data[1] (k) 2W
Data[j] (0) 2W	Data[j] (1) 2W	...	Data[j] (k) 2W
Super-Feature(1)			
Data[0] (0) 2W	Data[0] (1) 2W	...	Data[0] (k) 2W
Data[1] (0) 2W	Data[1] (1) 2W	...	Data[1] (k) 2W
Data[j] (0) 2W	Data[j] (1) 2W	...	Data[j] (k) 2W
Super-Feature(i)			
Data[0] (0) 2W	Data[0] (1) 2W	...	Data[0] (k) 2W
Data[1] (0) 2W	Data[1] (1) 2W	...	Data[1] (k) 2W
Data[j] (0) 2W	Data[j] (1) 2W	...	Data[j] (k) 2W

6.1.6.1.3 Immediate Fields Module

The immediate ID allows to change predefined fields through management commands without knowing their location in the NVM. The location is written in the module specified as follows:

Number of Immediate Fields (1W)					
Immediate ID (0) 1W	Pointer ¹ to Description Text 1W	Immediate Flags 1W	Mask 1W	Word Address ² LSB 1W	Word Address MSB 1W
Immediate ID (1) 1W	Pointer to Description Text 1W	Immediate Flags 1W	Mask 1W	Word Address LSB 1W	Word Address MSB 1W
...					
Immediate ID (i) ³ 1W	Pointer to Description Text 1W	Immediate Flags (i) ³ 1W	Mask 1W	Word Address LSB 1W	Word Address MSB 1W

1. The pointer is the word offset inside the metadata.
2. Word address is the word offset inside the metadata.
3. i=0...Number of immediate fields-1.

The immediate fields module flags are as follows:

Bit(s)	Content
15:5	Reserved
4	Always preserved ¹
3	Pattern=0
2	Hidden
1	Reserved
0	OEM ²

1. This flag indicates that this field is preserved in case of partial preservation.
2. Feature cannot be set by software as adaptive NVM feature. It is set during map generation per OEM.

6.1.6.1.4 Description Text Module

Description (0) Length 1W	X words description in ASCII
Description (1) Length 1W	Y words description in ASCII
...	...
Description (n) Length 1W	Z words description in ASCII

6.1.6.2 Feature Configuration

This section contains the current configurations chosen for the specific image. It is stored in the PFA with a *TLV Type* of 0x49.

Section Length 1W	FCID (0) 2W	FCID (1) 2W	...	FCID (n) 2W

Section Length is in word units, excluding the section length itself.

FCID is the feature configuration ID and consists of these four bytes:

- Feature Major #
- Feature Minor #
- Configuration Major #
- Configuration Minor #

FCID also represents the order of features in the adaptive NVM tool.

Notes:

- Super-features have FCID = 0xFXXX (0xFFFF value is reserved).
- Super-feature FCIDs should precede subordinated features FCIDs (or feature FCIDs placeholders).
- The feature configuration section can be finished by an end-of-list string of 0xFFFF + padding zeros.

6.1.6.3 Immediate Field Values

This section contains the current configurations chosen for the specific image that are immediate fields. It is stored in the PFA and pointed from address 0x4E.

Section Length is in word units and excludes the section length itself.

Immediate ID has 1 Word – Major # (1B) and Minor # (1B).

The immediate ID number should be unique on a per-project step.

- n=0...Number of immediate fields-1.
- The immediate field configuration section can be finished by an end-of-list string of 0xFFFF + padding zeros.

Section Length 1W	Immediate Field ID (0) 2W	Data (0) 1W	...	Immediate Field ID (n) 1W	Data (n) 1W

6.2 PLDM Header

6.2.1 PLDM Header Section Summary Table

This section is part of the PLDM package header. The PLDM header is not part of the NVM image

Word Address	Used By	Word Name	Reference
0x0000	HW	PackageHeaderIdentifier_0	Section 6.2.1.1
0x0001	HW	PackageHeaderIdentifier_1	Section 6.2.1.2
0x0002	HW	PackageHeaderIdentifier_2	Section 6.2.1.3
0x0003	HW	PackageHeaderIdentifier_3	Section 6.2.1.4
0x0004	HW	PackageHeaderIdentifier_4	Section 6.2.1.5
0x0005	HW	PackageHeaderIdentifier_5	Section 6.2.1.6
0x0006	HW	PackageHeaderIdentifier_6	Section 6.2.1.7
0x0007	HW	PackageHeaderIdentifier_7	Section 6.2.1.8
0x0008	HW	FormatRevision_HeaderSize_LSB	Section 6.2.1.9
0x0009	HW	Headersize_MSB	Section 6.2.1.10
0x000A	HW	ReleaseDateTime_0	Section 6.2.1.11
0x000B	HW	ReleaseDateTime_1	Section 6.2.1.12
0x000C	HW	ReleaseDateTime_2	Section 6.2.1.13
0x000D	HW	ReleaseDateTime_3	Section 6.2.1.14
0x000E	HW	ReleaseDateTime_4	Section 6.2.1.15
0x000F	HW	ReleaseDateTime_5	Section 6.2.1.16
0x0010	HW	ComponentBitmapBitLength	Section 6.2.1.17
0x0011	HW	PackageVersionStringType_Length	Section 6.2.1.18
0x0012	HW	PackageVersionString - FW component prefix	Section 6.2.1.19
0x0013	HW	PackageVersionString - FW component version 0	Section 6.2.1.20
0x0014	HW	PackageVersionString - FW component version 1	Section 6.2.1.21
0x0015	HW	PackageVersionString - FW component version 2	Section 6.2.1.22
0x0016	HW	PackageVersionString - FW component version 3	Section 6.2.1.23
0x0017	HW	PackageVersionString - OROM component prefix	Section 6.2.1.24
0x0018	HW	PackageVersionString - OROM component version 0	Section 6.2.1.25
0x0019	HW	PackageVersionString - OROM component version 1	Section 6.2.1.26
0x001A	HW	PackageVersionString - OROM component version 2	Section 6.2.1.27
0x001B	HW	PackageVersionString - OROM component version 3	Section 6.2.1.28
0x001C	HW	PackageVersionString - Netlist component prefix	Section 6.2.1.29
0x001D	HW	PackageVersionString - Netlist component version 0	Section 6.2.1.30
0x001E	HW	PackageVersionString - Netlist component version 1	Section 6.2.1.31
0x001F	HW	PackageVersionString - Netlist component version 2	Section 6.2.1.32
0x0020	HW	PackageVersionString - Netlist component version 3	Section 6.2.1.33
0x0021	HW	DeviceIDRecordCount and last byte of PackageVersionString	Section 6.2.1.34
0x0022	HW	Recordlength	Section 6.2.1.35
0x0023	HW	DescriptorCount and DeviceUpdateOptionFlags LSB	Section 6.2.1.36
0x0024	HW	DeviceUpdateOptionFlags - Middle	Section 6.2.1.37
0x0025	HW	DeviceUpdateOptionFlags - MSB and string type	Section 6.2.1.38

Word Address	Used By	Word Name	Reference
0x0026	HW	ComponentImageSetVersionStringLength and FirmwareDevicePackageDataLength - LSB	Section 6.2.1.39
0x0027	HW	FirmwareDevicePackageDataLength - MSB and ApplicableComponents	Section 6.2.1.40
0x0028	HW	ComponentImageSetVersionString - FW component prefix	Section 6.2.1.41
0x0029	HW	ComponentImageSetVersionString - FW component version 0	Section 6.2.1.42
0x002A	HW	ComponentImageSetVersionString - FW component version 1	Section 6.2.1.43
0x002B	HW	ComponentImageSetVersionString - FW component version 2	Section 6.2.1.44
0x002C	HW	ComponentImageSetVersionString - FW component version 3	Section 6.2.1.45
0x002D	HW	ComponentImageSetVersionString - OROM component prefix	Section 6.2.1.46
0x002E	HW	ComponentImageSetVersionString - OROM component version 0	Section 6.2.1.47
0x002F	HW	ComponentImageSetVersionString - OROM component version 1	Section 6.2.1.48
0x0030	HW	ComponentImageSetVersionString - OROM component version 2	Section 6.2.1.49
0x0031	HW	ComponentImageSetVersionString - OROM component version 3	Section 6.2.1.50
0x0032	HW	ComponentImageSetVersionString - Netlist component prefix	Section 6.2.1.51
0x0033	HW	ComponentImageSetVersionString - Netlist component version 0	Section 6.2.1.52
0x0034	HW	ComponentImageSetVersionString - Netlist component version 1	Section 6.2.1.53
0x0035	HW	ComponentImageSetVersionString - Netlist component version 2	Section 6.2.1.54
0x0036	HW	ComponentImageSetVersionString - Netlist component version 3	Section 6.2.1.55
0x0037	HW	ComponentImageSetVersionString - null padding	Section 6.2.1.56
0x0038	HW	InitialDescriptorType	Section 6.2.1.57
0x0039	HW	InitialDescriptorLength	Section 6.2.1.58
0x003A	HW	InitialDescriptorData	Section 6.2.1.59
0x003B	HW	AdditionalDescriptorType - DeviceID	Section 6.2.1.60
0x003C	HW	AdditionalDescriptorLength - DeviceID	Section 6.2.1.61
0x003D	HW	AdditionalDescriptorIdentifierData - Device ID	Section 6.2.1.62
0x003E	HW	AdditionalDescriptorType - SubVendorID	Section 6.2.1.63
0x003F	HW	AdditionalDescriptorLength - SubVendorID	Section 6.2.1.64
0x0040	HW	AdditionalDescriptorIdentifierData - SubVendorID	Section 6.2.1.65
0x0041	HW	AdditionalDescriptorType - SubSystemD	Section 6.2.1.66
0x0042	HW	AdditionalDescriptorLength - SubSystemD	Section 6.2.1.67
0x0043	HW	AdditionalDescriptorIdentifierData - SubSystemD	Section 6.2.1.68
0x0044	HW	FirmwareDevicePackageData - Header	Section 6.2.1.69
0x0045	HW	FirmwareDevicePackageData - GFID TLV Type	Section 6.2.1.70
0x0046	HW	FirmwareDevicePackageData - GFID TLV Length	Section 6.2.1.71
0x0047	HW	FirmwareDevicePackageData - GFID TLV Value - Current GFID IANA	Section 6.2.1.72
0x0048	HW	FirmwareDevicePackageData - GFID TLV Value - Current GFID deviceID	Section 6.2.1.73
0x0049	HW	FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz	Section 6.2.1.74
0x004A	HW	FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz0	Section 6.2.1.75
0x004B	HW	FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz1	Section 6.2.1.76
0x004C	HW	FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz2	Section 6.2.1.77
0x004D	HW	FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz3	Section 6.2.1.78
0x004E	HW	FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz4	Section 6.2.1.79
0x004F	HW	FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz5	Section 6.2.1.80
0x0050	HW	FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz6	Section 6.2.1.81
0x0051	HW	FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz7	Section 6.2.1.82

Word Address	Used By	Word Name	Reference
0x0052	HW	FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz8	Section 6.2.1.83
0x0053	HW	FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz9	Section 6.2.1.84
0x0054	HW	FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz10	Section 6.2.1.85
0x0055	HW	FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz11	Section 6.2.1.86
0x0056	HW	FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz12	Section 6.2.1.87
0x0057	HW	FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz13	Section 6.2.1.88
0x0058	HW	FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz14	Section 6.2.1.89
0x0059	HW	FirmwareDevicePackageData - GFID TLV Value - Original GFID IANA	Section 6.2.1.90
0x005A	HW	FirmwareDevicePackageData - GFID TLV Value - Original GFID deviceID	Section 6.2.1.91
0x005B	HW	FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz	Section 6.2.1.92
0x005C	HW	FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz0	Section 6.2.1.93
0x005D	HW	FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz1	Section 6.2.1.94
0x005E	HW	FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz2	Section 6.2.1.95
0x005F	HW	FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz3	Section 6.2.1.96
0x0060	HW	FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz4	Section 6.2.1.97
0x0061	HW	FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz5	Section 6.2.1.98
0x0062	HW	FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz6	Section 6.2.1.99
0x0063	HW	FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz7	Section 6.2.1.100
0x0064	HW	FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz8	Section 6.2.1.101
0x0065	HW	FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz9	Section 6.2.1.102
0x0066	HW	FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz10	Section 6.2.1.103
0x0067	HW	FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz11	Section 6.2.1.104
0x0068	HW	FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz12	Section 6.2.1.105
0x0069	HW	FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz13	Section 6.2.1.106
0x006A	HW	FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz14	Section 6.2.1.107
0x006B	HW	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Update type	Section 6.2.1.108
0x006C	HW	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Update length	Section 6.2.1.109
0x006D	HW	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - TLV Type	Section 6.2.1.110
0x006E	HW	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Update offset	Section 6.2.1.111
0x006F	HW	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Data length	Section 6.2.1.112
0x0070	HW	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - FW component prefix	Section 6.2.1.113
0x0071	HW	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - FW component version 0	Section 6.2.1.114
0x0072	HW	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - FW component version 1	Section 6.2.1.115
0x0073	HW	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - FW component version 2	Section 6.2.1.116
0x0074	HW	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - FW component version 3	Section 6.2.1.117
0x0075	HW	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - OROM component prefix	Section 6.2.1.118
0x0076	HW	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - OROM component version 0	Section 6.2.1.119
0x0077	HW	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - OROM component version 1	Section 6.2.1.120
0x0078	HW	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - OROM component version 2	Section 6.2.1.121

Word Address	Used By	Word Name	Reference
0x0079	HW	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - OROM component version 3	Section 6.2.1.122
0x007A	HW	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Netlist component prefix	Section 6.2.1.123
0x007B	HW	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Netlist component version 0	Section 6.2.1.124
0x007C	HW	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Netlist component version 1	Section 6.2.1.125
0x007D	HW	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Netlist component version 2	Section 6.2.1.126
0x007E	HW	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Netlist component version 3	Section 6.2.1.127
0x007F	HW	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - null padding	Section 6.2.1.128
0x0080	HW	FirmwareDevicePackageData - Additional TLVs - PXE TLV - Update Type	Section 6.2.1.129
0x0081	HW	FirmwareDevicePackageData - Additional TLVs - PXE TLV - Update Length	Section 6.2.1.130
0x0082	HW	FirmwareDevicePackageData - Additional TLVs - PXE TLV - TLV Type	Section 6.2.1.131
0x0083	HW	FirmwareDevicePackageData - Additional TLVs - PXE TLV - Update Offset	Section 6.2.1.132
0x0084	HW	FirmwareDevicePackageData - Additional TLVs - PXE TLV - Data Length	Section 6.2.1.133
0x0085	HW	FirmwareDevicePackageData - Additional TLVs - PXE TLV - Version	Section 6.2.1.134
0x0086	HW	FirmwareDevicePackageData - Additional TLVs - CIVD TLV - Update Type	Section 6.2.1.135
0x0087	HW	FirmwareDevicePackageData - Additional TLVs - CIVD TLV - Update Length	Section 6.2.1.136
0x0088	HW	FirmwareDevicePackageData - Additional TLVs - CIVD TLV - TLV Type	Section 6.2.1.137
0x0089	HW	FirmwareDevicePackageData - Additional TLVs - CIVD TLV - Update Offset	Section 6.2.1.138
0x008A	HW	FirmwareDevicePackageData - Additional TLVs - CIVD TLV - Data Length	Section 6.2.1.139
0x008B	HW	FirmwareDevicePackageData - Additional TLVs - CIVD TLV - Version High	Section 6.2.1.140
0x008C	HW	FirmwareDevicePackageData - Additional TLVs - CIVD TLV - Version Low	Section 6.2.1.141
0x008D	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Update Type	Section 6.2.1.142
0x008E	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Update Length	Section 6.2.1.143
0x008F	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - TLV Type	Section 6.2.1.144
0x0090	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Update Offset	Section 6.2.1.145
0x0091	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Data Length	Section 6.2.1.146
0x0092	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Dell identifier	Section 6.2.1.147
0x0093	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - CEM Flags	Section 6.2.1.148
0x0094	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Number of Inventory NVM Entries	Section 6.2.1.149
0x0095	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Media Type ID Length	Section 6.2.1.150
0x0096	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Media Type Data	Section 6.2.1.151
0x0097	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Number of Extended Capabilities Entries	Section 6.2.1.152
0x0098	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Capabilities Flags ID Length	Section 6.2.1.153
0x0099	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Capabilities Flags Data High	Section 6.2.1.154
0x009A	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Capabilities Flags Data Low	Section 6.2.1.155

Word Address	Used By	Word Name	Reference
0x009B	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - DCB Capabilities ID Length	Section 6.2.1.156
0x009C	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - DCB Capabilities Data	Section 6.2.1.157
0x009D	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - E-switch Capabilities ID Length	Section 6.2.1.158
0x009E	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - E-switch Capabilities Data	Section 6.2.1.159
0x009F	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - NIC Partitioning Capabilities ID Length	Section 6.2.1.160
0x00A0	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - NIC Partitioning Capabilities Data	Section 6.2.1.161
0x00A1	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Number of Partition Information NVM Entries	Section 6.2.1.162
0x00A2	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality ID Length	Section 6.2.1.163
0x00A3	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 1+2	Section 6.2.1.164
0x00A4	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 3+4	Section 6.2.1.165
0x00A5	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 5+6	Section 6.2.1.166
0x00A6	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 7+8	Section 6.2.1.167
0x00A7	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 9+10	Section 6.2.1.168
0x00A8	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 11+12	Section 6.2.1.169
0x00A9	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 13+14	Section 6.2.1.170
0x00AA	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 15+16	Section 6.2.1.171
0x00AB	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 17+18	Section 6.2.1.172
0x00AC	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 19+20	Section 6.2.1.173
0x00AD	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 21+22	Section 6.2.1.174
0x00AE	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 23+24	Section 6.2.1.175
0x00AF	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 25+26	Section 6.2.1.176
0x00B0	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 27+28	Section 6.2.1.177
0x00B1	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 29+30	Section 6.2.1.178
0x00B2	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 30+31	Section 6.2.1.179
0x00B3	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 32+33	Section 6.2.1.180
0x00B4	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 34+35	Section 6.2.1.181
0x00B5	HW	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 36+37	Section 6.2.1.182
0x00B6	HW	FirmwareDevicePackageData - Additional TLVs - VPD update - Update Type	Section 6.2.1.183

Word Address	Used By	Word Name	Reference
0x00B7	HW	FirmwareDevicePackageData - Additional TLVs - VPD update - Update Length	Section 6.2.1.184
0x00B8	HW	FirmwareDevicePackageData - Additional TLVs - VPD update - VPD V0 key	Section 6.2.1.185
0x00B9	HW	FirmwareDevicePackageData - Additional TLVs - VPD update - Data length	Section 6.2.1.186
0x00BA	HW	FirmwareDevicePackageData - Additional TLVs - VPD update - V0 key data0	Section 6.2.1.187
0x00BB	HW	FirmwareDevicePackageData - Additional TLVs - VPD update - V0 key data 1	Section 6.2.1.188
0x00BC	HW	FirmwareDevicePackageData - Additional TLVs - VPD update - V0 key data 2	Section 6.2.1.189
0x00BD	HW	FirmwareDevicePackageData - Additional TLVs - VPD update - V0 key data 3	Section 6.2.1.190
0x00BE	HW	FirmwareDevicePackageData - Additional TLVs - VPD update - V0 key data 4	Section 6.2.1.191
0x00BF	HW	FirmwareDevicePackageData - Additional TLVs - VPD update - V0 key data 5	Section 6.2.1.192
0x00C0	HW	FirmwareDevicePackageData - Additional TLVs - IBA TLV - Update Type	Section 6.2.1.193
0x00C1	HW	FirmwareDevicePackageData - Additional TLVs - IBA TLV - Update Length	Section 6.2.1.194
0x00C2	HW	FirmwareDevicePackageData - Additional TLVs - IBA TLV - TLV Type	Section 6.2.1.195
0x00C3	HW	FirmwareDevicePackageData - Additional TLVs - IBA TLV - Update Offset	Section 6.2.1.196
0x00C4	HW	FirmwareDevicePackageData - Additional TLVs - IBA TLV - Data Length	Section 6.2.1.197
0x00C5	HW	FirmwareDevicePackageData - Additional TLVs - IBA TLV - Version	Section 6.2.1.198
0x00C6	HW	ComponentImageCount	Section 6.2.1.199
0x00C7	HW	ComponentClassification	Section 6.2.1.200
0x00C8	HW	ComponentIdentifier	Section 6.2.1.201
0x00C9	HW	ComponentComparisonStamp LSB	Section 6.2.1.202
0x00CA	HW	ComponentComparisonStamp MSB	Section 6.2.1.203
0x00CB	HW	ComponentOptions	Section 6.2.1.204
0x00CC	HW	RequestedComponentActivationMethod	Section 6.2.1.205
0x00CD	HW	ComponentLocationOffset LSB	Section 6.2.1.206
0x00CE	HW	ComponentLocationOffset MSB	Section 6.2.1.207
0x00CF	HW	ComponentSize LSB	Section 6.2.1.208
0x00D0	HW	ComponentSize MSB	Section 6.2.1.209
0x00D1	HW	ComponentVersionStringTypeAndLength	Section 6.2.1.210
0x00D2	HW	ComponentVersionString - Dev starter Major	Section 6.2.1.211
0x00D3	HW	ComponentVersionString - Dev starter Minor	Section 6.2.1.212
0x00D4	HW	ComponentVersionString - EETRACK-ID MSB	Section 6.2.1.213
0x00D5	HW	ComponentVersionString - EETRACK-ID LSB	Section 6.2.1.214
0x00D6	HW	ComponentVersionString - Dot and Srev Byte7	Section 6.2.1.215
0x00D7	HW	ComponentVersionString - Srev Bytes 6-5	Section 6.2.1.216
0x00D8	HW	ComponentVersionString - Srev Bytes 4-3	Section 6.2.1.217
0x00D9	HW	ComponentVersionString - Srev Bytes 2-1	Section 6.2.1.218
0x00DA	HW	ComponentVersionString - Srev Byte 0 and null	Section 6.2.1.219
0x00DB	HW	ComponentClassification	Section 6.2.1.220
0x00DC	HW	ComponentIdentifier	Section 6.2.1.221
0x00DD	HW	ComponentComparisonStamp LSB	Section 6.2.1.222
0x00DE	HW	ComponentComparisonStamp MSB	Section 6.2.1.223
0x00DF	HW	ComponentOptions	Section 6.2.1.224
0x00E0	HW	RequestedComponentActivationMethod	Section 6.2.1.225
0x00E1	HW	ComponentLocationOffset LSB	Section 6.2.1.226
0x00E2	HW	ComponentLocationOffset MSB	Section 6.2.1.227
0x00E3	HW	ComponentSize LSB	Section 6.2.1.228

Word Address	Used By	Word Name	Reference
0x00E4	HW	ComponentSize MSB	Section 6.2.1.229
0x00E5	HW	ComponentVersionStringTypeAndLength	Section 6.2.1.230
0x00E6	HW	ComponentVersionString - CVID High MSB	Section 6.2.1.231
0x00E7	HW	ComponentVersionString - CVID High LSB	Section 6.2.1.232
0x00E8	HW	ComponentVersionString - CVID Low MSB	Section 6.2.1.233
0x00E9	HW	ComponentVersionString - CVID Low LSB	Section 6.2.1.234
0x00EA	HW	ComponentVersionString - Dot and Srev Byte7	Section 6.2.1.235
0x00EB	HW	ComponentVersionString - Srev Bytes 6-5	Section 6.2.1.236
0x00EC	HW	ComponentVersionString - Srev Bytes 4-3	Section 6.2.1.237
0x00ED	HW	ComponentVersionString - Srev Bytes 2-1	Section 6.2.1.238
0x00EE	HW	ComponentVersionString - Srev Byte 0 and null	Section 6.2.1.239
0x00EF	HW	ComponentClassification	Section 6.2.1.240
0x00F0	HW	ComponentIdentifier	Section 6.2.1.241
0x00F1	HW	ComponentComparisonStamp LSB	Section 6.2.1.242
0x00F2	HW	ComponentComparisonStamp MSB	Section 6.2.1.243
0x00F3	HW	ComponentOptions	Section 6.2.1.244
0x00F4	HW	RequestedComponentActivationMethod	Section 6.2.1.245
0x00F5	HW	ComponentLocationOffset LSB	Section 6.2.1.246
0x00F6	HW	ComponentLocationOffset MSB	Section 6.2.1.247
0x00F7	HW	ComponentSize LSB	Section 6.2.1.248
0x00F8	HW	ComponentSize MSB	Section 6.2.1.249
0x00F9	HW	ComponentVersionStringTypeAndLength	Section 6.2.1.250
0x00FA	HW	ComponentVersionString - ReleaseVersion Major Bytes 7-6	Section 6.2.1.251
0x00FB	HW	ComponentVersionString - ReleaseVersion Major Bytes 5-4	Section 6.2.1.252
0x00FC	HW	ComponentVersionString - ReleaseVersion Major Bytes 3-2	Section 6.2.1.253
0x00FD	HW	ComponentVersionString - ReleaseVersion Major Bytes 1-0	Section 6.2.1.254
0x00FE	HW	ComponentVersionString - Dot and ReleaseVersion Minor Byte 7	Section 6.2.1.255
0x00FF	HW	ComponentVersionString - ReleaseVersion Minor Bytes 6-5	Section 6.2.1.256
0x0100	HW	ComponentVersionString - ReleaseVersion Minor Bytes 4-3	Section 6.2.1.257
0x0101	HW	ComponentVersionString - ReleaseVersion Minor Bytes 2-1	Section 6.2.1.258
0x0102	HW	ComponentVersionString - ReleaseVersion Minor Byte 0 and Dot	Section 6.2.1.259
0x0103	HW	ComponentVersionString - ReleaseVersion Type Bytes 7-6	Section 6.2.1.260
0x0104	HW	ComponentVersionString - ReleaseVersion Type Bytes 5-4	Section 6.2.1.261
0x0105	HW	ComponentVersionString - ReleaseVersion Type Bytes 3-2	Section 6.2.1.262
0x0106	HW	ComponentVersionString - ReleaseVersion Type Bytes 0-1	Section 6.2.1.263
0x0107	HW	ComponentVersionString - Dot and Customer Netlist IANA Byte 7	Section 6.2.1.264
0x0108	HW	ComponentVersionString - Customer Netlist IANA Bytes 6-5	Section 6.2.1.265
0x0109	HW	ComponentVersionString - Customer Netlist IANA Bytes 4-3	Section 6.2.1.266
0x010A	HW	ComponentVersionString - Customer Netlist IANA Bytes 2-1	Section 6.2.1.267
0x010B	HW	ComponentVersionString - Customer Netlist IANA Byte 0 and Dot	Section 6.2.1.268
0x010C	HW	ComponentVersionString - Customer Netlist Version Bytes 3-2	Section 6.2.1.269
0x010D	HW	ComponentVersionString - Customer Netlist Version Bytes 1-0	Section 6.2.1.270
0x010E	HW	ComponentVersionString - nulls	Section 6.2.1.271
0x010F	HW	PackageHeaderChecksum - LSB	Section 6.2.1.272
0x0110	HW	PackageHeaderChecksum - MSB	Section 6.2.1.273

6.2.1.1 PackageHeaderIdentifier_0 - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	PackageHeaderIdentifier	0x18F0	

6.2.1.2 PackageHeaderIdentifier_1 - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	PackageHeaderIdentifier	0x8C87	

6.2.1.3 PackageHeaderIdentifier_2 - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	PackageHeaderIdentifier	0x7DCB	

6.2.1.4 PackageHeaderIdentifier_3 - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	PackageHeaderIdentifier	0x4349	

6.2.1.5 PackageHeaderIdentifier_4 - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	PackageHeaderIdentifier	0x0098	

6.2.1.6 PackageHeaderIdentifier_5 - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	PackageHeaderIdentifier	0x2FA0	

6.2.1.7 PackageHeaderIdentifier_6 - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	PackageHeaderIdentifier	0x9A05	

6.2.1.8 PackageHeaderIdentifier_7 - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	PackageHeaderIdentifier	0x02CA	

6.2.1.9 FormatRevision_HeaderSize_LSB - 0x0008

Bits	Field Name	Default NVM Value	Description
15:8	HeaderSize_LSB		
7:0	FormatRevision	0x01	

6.2.1.10 Headersize_MSB - 0x0009

Bits	Field Name	Default NVM Value	Description
15:8	ReleaseDateTimeUTC_offset_LSB	0x0	timestamp104 byte 0
7:0	Headersize_MSB		

6.2.1.11 ReleaseDateTime_0 - 0x000A

Bits	Field Name	Default NVM Value	Description
15:8	ReleaseDateTimeMsec_0	0x0	timestamp104 byte 2
7:0	ReleaseDateTimeUTC_offset_MSB_by	0x0	timestamp104 byte 1

6.2.1.12 ReleaseDateTime_1 - 0x000B

Bits	Field Name	Default NVM Value	Description
15:0	ReleaseDateTimeMsec_1_2	0x0	timestamp104 byte 3:4

6.2.1.13 ReleaseDateTime_2 - 0x000C

Bits	Field Name	Default NVM Value	Description
15:8	ReleaseDateTimeMin	0x0	timestamp104 byte 6
7:0	ReleaseDateTimeSec	0x0	timestamp104 byte 5

6.2.1.14 ReleaseDateTime_3 - 0x000D

Bits	Field Name	Default NVM Value	Description
15:8	ReleaseDateTimeDay	0x0	timestamp104 byte 8
7:0	ReleaseDateTimeHour	0x0	timestamp104 byte 7

6.2.1.15 ReleaseDateTime_4 - 0x000E

Bits	Field Name	Default NVM Value	Description
15:8	ReleaseDateTimeYear_LSB	0x0	timestamp104 byte 10
7:0	ReleaseDateTimeMonth	0x0	timestamp104 byte 9

6.2.1.16 ReleaseDateTime_5 - 0x000F

Bits	Field Name	Default NVM Value	Description
15:12	ReleaseDateTimeUTC Resolution	0x2	timestamp104 byte 12 [7:4] - resolution is minutes
11:8	ReleaseDateTimeResolution	0x7	timestamp104 byte 12 [3:0] - resolution is minutes
7:0	ReleaseDateTimeYear_MSB	0x0	timestamp104 byte 11

6.2.1.17 ComponentBitmapBitLength - 0x0010

Bits	Field Name	Default NVM Value	Description
15:0	ComponentBitmapBitLength	0x8	

6.2.1.18 PackageVersionStringType_Length - 0x0011

Bits	Field Name	Default NVM Value	Description
15:8	PackageVersionStringLength	0x1F	Assuming a string length of 31 - the last byte is part of the Firmware Device Identification Area
7:0	PackageVersionStringType	0x1	

6.2.1.19 PackageVersionString - FW component prefix - 0x0012

Bits	Field Name	Default NVM Value	Description
15:0	PackageVersionString - FW component prefix	0x3A4E	"N:" in ASCII

6.2.1.20 PackageVersionString - FW component version 0 - 0x0013

Bits	Field Name	Default NVM Value	Description
15:0	PackageVersionString - FW component version 0	0x3030	FW component version 0-3: ASCII presentation of FW comparison stamp. Values are zero (in ASCII) padded.

6.2.1.21 PackageVersionString - FW component version 1 - 0x0014

Bits	Field Name	Default NVM Value	Description
15:0	PackageVersionString - FW component version 1	0x3030	

6.2.1.22 PackageVersionString - FW component version 2 - 0x0015

Bits	Field Name	Default NVM Value	Description
15:0	PackageVersionString - FW component version 2	0x3030	

6.2.1.23 PackageVersionString - FW component version 3 - 0x0016

Bits	Field Name	Default NVM Value	Description
15:0	PackageVersionString - FW component version 3	0x3030	

6.2.1.24 PackageVersionString - OROM component prefix - 0x0017

Bits	Field Name	Default NVM Value	Description
15:0	PackageVersionString - OROM component prefix	0x3A4F	"O:" in ASCII

6.2.1.25 PackageVersionString - OROM component version 0 - 0x0018

Bits	Field Name	Default NVM Value	Description
15:0	PackageVersionString - OROM component version 0	0x3030	OROM component version 0-3: ASCII presentation of OROM comparison stamp. Values are zero (in ASCII) padded.

6.2.1.26 PackageVersionString - OROM component version 1 - 0x0019

Bits	Field Name	Default NVM Value	Description
15:0	PackageVersionString - OROM component version 1	0x3030	

6.2.1.27 PackageVersionString - OROM component version 2 - 0x001A

Bits	Field Name	Default NVM Value	Description
15:0	PackageVersionString - OROM component version 2	0x3030	

6.2.1.28 PackageVersionString - OROM component version 3 - 0x001B

Bits	Field Name	Default NVM Value	Description
15:0	PackageVersionString - OROM component version 3	0x3030	

6.2.1.29 PackageVersionString - Netlist component prefix - 0x001C

Bits	Field Name	Default NVM Value	Description
15:0	PackageVersionString - Netlist component prefix	0x3A54	"T:" in ASCII

6.2.1.30 PackageVersionString - Netlist component version 0 - 0x001D

Bits	Field Name	Default NVM Value	Description
15:0	PackageVersionString - Netlist component version 0	0x3030	Netlist component version 0-3: ASCII presentation of Netlist comparison stamp. Values are zero (in ASCII) padded.

6.2.1.31 PackageVersionString - Netlist component version 1 - 0x001E

Bits	Field Name	Default NVM Value	Description
15:0	PackageVersionString - Netlist component version 1	0x3030	

6.2.1.32 PackageVersionString - Netlist component version 2 - 0x001F

Bits	Field Name	Default NVM Value	Description
15:0	PackageVersionString - Netlist component version 2	0x3030	

6.2.1.33 PackageVersionString - Netlist component version 3 - 0x0020

Bits	Field Name	Default NVM Value	Description
15:0	PackageVersionString - Netlist component version 3	0x3030	

6.2.1.34 DeviceIDRecordCount and last byte of PackageVersionString - 0x0021

Bits	Field Name	Default NVM Value	Description
15:8	DeviceIDRecordCount	0x1	
7:0	PackageVersionString last byte	0x0	

6.2.1.35 Recordlength - 0x0022

Bits	Field Name	Default NVM Value	Description
15:0	RecordLength		Length in: 1 Byte unit First Section -> Word: PLDM Header -> Recordlength Last Section -> Word: PLDM Header -> FirmwareDevicePackageData - Additional TLVs - IBA TLV - Version The total length in bytes for this record. The length shall include the RecordLength, DescriptorCount, DeviceUpdateOptionFlags, ComponentImageSetVersionStringType, ComponentSetVersionStringLength, FirmwareDevicePackageDataLength, ApplicableComponents, ComponentImageSetVersionString, RecordDescriptors, and FirmwareDevicePackageData fields.

6.2.1.36 DescriptorCount and DeviceUpdateOptionFlags LSB - 0x0023

Bits	Field Name	Default NVM Value	Description
15:9	DeviceUpdateOptionFlags LSB	0x0	bits [31:24] - Reserved
8	DeviceUpdateOptionFlags LSB - Continue component updates after failure	0x0	bits [31:24] - Reserved
7:0	DescriptorCount	0x4	The number of descriptors included within the RecordDescriptors field for this record.

6.2.1.37 DeviceUpdateOptionFlags - Middle - 0x0024

Bits	Field Name	Default NVM Value	Description
15:0	DeviceUpdateOptionFlags - Middlelegs - Middle	0x0	Bits [23:8] Reserved

6.2.1.38 DeviceUpdateOptionFlags - MSB and string type - 0x0025

Bits	Field Name	Default NVM Value	Description
15:8	ComponentImageSetVersionStringType	0x1	Valid values are: 0x0 = Unknown 0x1 = ASCII 0x2 = UTF-8 0x3 = UTF-16 0x4 = UTF-16LE 0x5 = UTF-16BE
7:0	DeviceUpdateOptionFlags - MSB	0x0	

6.2.1.39 ComponentImageSetVersionStringLength and FirmwareDevicePackageDataLength - LSB - 0x0026

Bits	Field Name	Default NVM Value	Description
15:8	FirmwareDevicePackageDataLength - LSB		
7:0	ComponentImageSetVersionStringLength	0x20	

6.2.1.40 FirmwareDevicePackageDataLength - MSB and ApplicableComponents - 0x0027

Bits	Field Name	Default NVM Value	Description
15:11	Reserved ApplicableComponents	0x0	
10	FW Link topology ApplicableComponent	0x0	Valid values are: 0x0 = Non Applicable 0x1 = Applicable
9	OROM ApplicableComponent	0x1	Valid values are: 0x0 = Non Applicable 0x1 = Applicable
8	NVM ApplicableComponent	0x1	Valid values are: 0x0 = Non Applicable 0x1 = Applicable
7:0	FirmwareDevicePackageDataLength -MSB		

6.2.1.41 ComponentImageSetVersionString - FW component prefix - 0x0028

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - FW component prefix	0x3A4E	"N:" in ASCII

6.2.1.42 ComponentImageSetVersionString - FW component version 0 - 0x0029

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - FW component version 0	0x3030	FW component version 0-3: ASCII presentation of FW comparison stamp. Values are zero (in ASCII) padded.

6.2.1.43 ComponentImageSetVersionString - FW component version 1 - 0x002A

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - FW component version 1	0x3030	

6.2.1.44 ComponentImageSetVersionString - FW component version 2 - 0x002B

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - FW component version 2	0x3030	

6.2.1.45 ComponentImageSetVersionString - FW component version 3 - 0x002C

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - FW component version 3	0x3030	

6.2.1.46 ComponentImageSetVersionString - OROM component prefix - 0x002D

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - OROM component prefix	0x3A4F	"O:" in ASCII

6.2.1.47 ComponentImageSetVersionString - OROM component version 0 - 0x002E

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - OROM component version 0	0x3030	OROM component version 0-3: ASCII presentation of OROM comparison stamp. Values are zero (in ASCII) padded.

6.2.1.48 ComponentImageSetVersionString - OROM component version 1 - 0x002F

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - OROM component version 1	0x3030	

6.2.1.49 ComponentImageSetVersionString - OROM component version 2 - 0x0030

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - OROM component version 2	0x3030	

6.2.1.50 ComponentImageSetVersionString - OROM component version 3 - 0x0031

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - OROM component version 3	0x3030	

6.2.1.51 ComponentImageSetVersionString - Netlist component prefix - 0x0032

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - Netlist component prefix	0x3A54	"T:" in ASCII

6.2.1.52 ComponentImageSetVersionString - Netlist component version 0 - 0x0033

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - Netlist component version 0	0x3030	Netlist componenet version 0-3: ASCII presentation of Netlist comparison stamp. Values are zero (in ASCII) padded.

6.2.1.53 ComponentImageSetVersionString - Netlist component version 1 - 0x0034

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - Netlist component version 1	0x3030	

6.2.1.54 ComponentImageSetVersionString - Netlist component version 2 - 0x0035

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - Netlist component version 2	0x3030	

6.2.1.55 ComponentImageSetVersionString - Netlist component version 3 - 0x0036

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - Netlist component version 3	0x3030	

6.2.1.56 ComponentImageSetVersionString - null padding - 0x0037

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - null padding	0x0	

6.2.1.57 InitialDescriptorType - 0x0038

Bits	Field Name	Default NVM Value	Description
15:0	InitialDescriptorType	0x0	Indicates the type of the Initial descriptor. The initial descriptor for a device shall be defined by one of the following (PCI Vendor ID, IANA Enterprise ID, UUID, PnP Vendor ID, or ACPI Vendor ID). Valid values are: 0x0 = PCI Vendor ID 0x1 = IANA Enterprise ID 0x2 = UUID 0x3 = PnP Vendor ID 0x4 = ACPI Vendor ID 0x100 = PCI Device ID 0x101 = PCI Subsystem Vendor ID 0x102 = PCI Subsystem ID 0x103 = PCI Revision ID 0x104 = PnP Product Identifier 0x105 = ACPI Product Identifier 0xFFFF = Vendor Defined

6.2.1.58 InitialDescriptorLength - 0x0039

Bits	Field Name	Default NVM Value	Description
15:0	InitialDescriptorLength	0x2	Indicates the length, in bytes, of the InitialDescriptorData field.

6.2.1.59 InitialDescriptorData - 0x003A

Bits	Field Name	Default NVM Value	Description
15:0	VendorID	0x8086	PCI Vendor ID assigned to the FD vendor.

6.2.1.60 AdditionalDescriptorType - DeviceID - 0x003B

Bits	Field Name	Default NVM Value	Description
15:0	AdditionalDescriptorType	0x100	Indicates the type of the Initial descriptor. The initial descriptor for a device shall be defined by one of the following (PCI Vendor ID, IANA Enterprise ID, UUID, PnP Vendor ID, or ACPI Vendor ID). Valid values are: 0x0 = PCI Vendor ID 0x1 = IANA Enterprise ID 0x2 = UUID 0x3 = PnP Vendor ID 0x4 = ACPI Vendor ID 0x100 = PCI Device ID 0x101 = PCI Subsystem Vendor ID 0x102 = PCI Subsystem ID 0x103 = PCI Revision ID 0x104 = PnP Product Identifier 0x105 = ACPI Product Identifier 0xFFFF = Vendor Defined

6.2.1.61 AdditionalDescriptorLength - DeviceID - 0x003C

Bits	Field Name	Default NVM Value	Description
15:0	AdditionalDescriptorLength	0x2	Indicates the length, in bytes, of the InitialDescriptorData field.

6.2.1.62 AdditionalDescriptorIdentifierData - Device ID - 0x003D

Bits	Field Name	Default NVM Value	Description
15:0	DeviceID	0x0	PCI Vendor ID assigned to the FD vendor.

6.2.1.63 AdditionalDescriptorType - SubVendorID - 0x003E

Bits	Field Name	Default NVM Value	Description
15:0	AdditionalDescriptorType	0x101	Indicates the type of the Initial descriptor. The initial descriptor for a device shall be defined by one of the following (PCI Vendor ID, IANA Enterprise ID, UUID, PnP Vendor ID, or ACPI Vendor ID). Valid values are: 0x0 = PCI Vendor ID 0x1 = IANA Enterprise ID 0x2 = UUID 0x3 = PnP Vendor ID 0x4 = ACPI Vendor ID 0x100 = PCI Device ID 0x101 = PCI Subsystem Vendor ID 0x102 = PCI Subsystem ID 0x103 = PCI Revision ID 0x104 = PnP Product Identifier 0x105 = ACPI Product Identifier 0xFFFF = Vendor Defined

6.2.1.64 AdditionalDescriptorLength - SubVendorID - 0x003F

Bits	Field Name	Default NVM Value	Description
15:0	AdditionalDescriptorLength	0x2	Indicates the length, in bytes, of the InitialDescriptorData field.

6.2.1.65 AdditionalDescriptorIdentifierData - SubVendorID - 0x0040

Bits	Field Name	Default NVM Value	Description
15:0	SubVendorID	0x0	PCI Vendor ID assigned to the FD vendor.

6.2.1.66 AdditionalDescriptorType - SubSystemD - 0x0041

Bits	Field Name	Default NVM Value	Description
15:0	AdditionalDescriptorType	0x102	Indicates the type of the Initial descriptor. The initial descriptor for a device shall be defined by one of the following (PCI Vendor ID, IANA Enterprise ID, UUID, PnP Vendor ID, or ACPI Vendor ID). Valid values are: 0x0 = PCI Vendor ID 0x1 = IANA Enterprise ID 0x2 = UUID 0x3 = PnP Vendor ID 0x4 = ACPI Vendor ID 0x100 = PCI Device ID 0x101 = PCI Subsystem Vendor ID 0x102 = PCI Subsystem ID 0x103 = PCI Revision ID 0x104 = PnP Product Identifier 0x105 = ACPI Product Identifier 0xFFFF = Vendor Defined

6.2.1.67 AdditionalDescriptorLength - SubSystemD - 0x0042

Bits	Field Name	Default NVM Value	Description
15:0	AdditionalDescriptorLength	0x2	Indicates the length, in bytes, of the InitialDescriptorData field.

6.2.1.68 AdditionalDescriptorIdentifierData - SubSystemD - 0x0043

Bits	Field Name	Default NVM Value	Description
15:0	SubSystemD	0x0	PCI Vendor ID assigned to the FD vendor.

6.2.1.69 FirmwareDevicePackageData - Header - 0x0044

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:2	Reserved	0x0	
1	O-ROM version to be set in VPD	0x0	0 = do not set O-ROM version in VPD 1 = Set O-ROM version in VPD
0	PFA Preserve	0x0	PFA Preserve (default mode). Should be set to zero

6.2.1.70 FirmwareDevicePackageData - GFID TLV Type - 0x0045

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV- Type	0x0001	CURRENT_GFID

6.2.1.71 FirmwareDevicePackageData - GFID TLV Length - 0x0046

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Length		

6.2.1.72 FirmwareDevicePackageData - GFID TLV Value - Current GFID IANA - 0x0047

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value-IANA	0xB85C	Valid values are: 0x0009 = Cisco 0x0157 = Intel 0x02A2 = Dell 0x4A66 = Lenovo 0xB85C = HPe

6.2.1.73 FirmwareDevicePackageData - GFID TLV Value - Current GFID deviceID - 0x0048

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value - device ID	0x57AC	Valid values are: 0x1590 = CVL 0x1598 = CVL-SD 0x1888 = CPK 0x1895 = CPK-BMSM 0x1896 = CPK-BTS 0x57AC = LKV

6.2.1.74 FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz - 0x0049

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.75 FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz0 - 0x004A

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.76 FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz1 - 0x004B

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.77 FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz2 - 0x004C

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.78 FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz3 - 0x004D

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.79 FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz4 - 0x004E

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.80 FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz5 - 0x004F

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.81 FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz6 - 0x0050

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.82 FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz7 - 0x0051

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.83 FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz8 - 0x0052

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.84 FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz9 - 0x0053

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.85 FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz10 - 0x0054

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.86 FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz11 - 0x0055

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.87 FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz12 - 0x0056

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.88 FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz13 - 0x0057

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.89 FirmwareDevicePackageData - GFID TLV Value - Current GFID zeroz14 - 0x0058

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.90 FirmwareDevicePackageData - GFID TLV Value - Original GFID IANA - 0x0059

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value-IANA	0xB85C	Valid values are: 0x0009 = Cisco 0x0157 = Intel 0x02A2 = Dell 0x4A66 = Lenovo 0xB85C = HPe

6.2.1.91 FirmwareDevicePackageData - GFID TLV Value - Original GFID deviceID - 0x005A

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value - device ID	0x57AC	Valid values are: 0x1590 = CVL 0x1598 = CVL-SD 0x1888 = CPK 0x1895 = CPK-BMSM 0x1896 = CPK-BTS 0x57AC = LKV

6.2.1.92 FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz - 0x005B

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.93 FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz0 - 0x005C

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.94 FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz1 - 0x005D

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.95 FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz2 - 0x005E

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.96 FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz3 - 0x005F

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.97 FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz4 - 0x0060

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.98 FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz5 - 0x0061

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.99 FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz6 - 0x0062

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.100 FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz7 - 0x0063

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.101 FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz8 - 0x0064

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.102 FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz9 - 0x0065

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.103 FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz10 - 0x0066

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.104 FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz11 - 0x0067

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.105 FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz12 - 0x0068

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.106 FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz13 - 0x0069

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.107 FirmwareDevicePackageData - GFID TLV Value - Original GFID zeroz14 - 0x006A

An optional data field that can be provided within the firmware update package which the UA shall transfer to the FD during the firmware update process. The UA has no knowledge of what data is contained within this field, and will simply pass the contents of this field when the FD requests it via the GetPackageData command response.

If the FirmwareDevicePackageDataLength field is set to 0x0000 then this field contains no data and is zero bytes in length.

Bits	Field Name	Default NVM Value	Description
15:0	TLV Value	0x0	

6.2.1.108 FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Update type - 0x006B

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Update type	0x10	

6.2.1.109 FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Update length - 0x006C

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Update length	0x13	

6.2.1.110 FirmwareDevicePackageData - Additional TLVs - PLDM TLV - TLV Type - 0x006D

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - TLV Type	0x127	

6.2.1.111 FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Update offset - 0x006E

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Update offset	0x0	

6.2.1.112 FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Data length - 0x006F

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Data length	0x10	

6.2.1.113 FirmwareDevicePackageData - Additional TLVs - PLDM TLV - FW component prefix - 0x0070

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - FW component prefix	0x3A4E	"N:" in ASCII

6.2.1.114 FirmwareDevicePackageData - Additional TLVs - PLDM TLV - FW component version 0 - 0x0071

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - FW component version 0	0x3030	FW component version 0-3: ASCII presentation of FW comparison stamp. Values are zero (in ASCII) padded.

6.2.1.115 FirmwareDevicePackageData - Additional TLVs - PLDM TLV - FW component version 1 - 0x0072

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - FW component version 1	0x3030	

6.2.1.116 FirmwareDevicePackageData - Additional TLVs - PLDM TLV - FW component version 2 - 0x0073

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - FW component version 2	0x3030	

6.2.1.117 FirmwareDevicePackageData - Additional TLVs - PLDM TLV - FW component version 3 - 0x0074

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - FW component version 3	0x3030	

6.2.1.118 FirmwareDevicePackageData - Additional TLVs - PLDM TLV - OROM component prefix - 0x0075

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - OROM component prefix	0x3A4F	"O:" in ASCII

6.2.1.119 FirmwareDevicePackageData - Additional TLVs - PLDM TLV - OROM component version 0 - 0x0076

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - OROM component version 0	0x3030	OROM component version 0-3: ASCII presentation of OROM comparison stamp. Values are zero (in ASCII) padded.

6.2.1.120 FirmwareDevicePackageData - Additional TLVs - PLDM TLV - OROM component version 1 - 0x0077

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - OROM component version 1	0x3030	

6.2.1.121 FirmwareDevicePackageData - Additional TLVs - PLDM TLV - OROM component version 2 - 0x0078

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - OROM component version 2	0x3030	

6.2.1.122 FirmwareDevicePackageData - Additional TLVs - PLDM TLV - OROM component version 3 - 0x0079

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - OROM component version 3	0x3030	

6.2.1.123 FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Netlist component prefix - 0x007A

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Netlist component prefix	0x3A54	"T:" in ASCII

6.2.1.124 FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Netlist component version 0 - 0x007B

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Netlist component version 0	0x3030	Netlist component version 0-3: ASCII presentation of Netlist comparison stamp. Values are zero (in ASCII) padded.

6.2.1.125 FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Netlist component version 1 - 0x007C

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Netlist component version 1	0x3030	

6.2.1.126 FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Netlist component version 2 - 0x007D

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Netlist component version 2	0x3030	

6.2.1.127 FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Netlist component version 3 - 0x007E

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - Netlist component version 3	0x3030	

6.2.1.128 FirmwareDevicePackageData - Additional TLVs - PLDM TLV - null padding - 0x007F

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - null padding	0x0	

6.2.1.129 FirmwareDevicePackageData - Additional TLVs - PXE TLV - Update Type - 0x0080

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PXE TLV - Update type	0x10	

6.2.1.130 FirmwareDevicePackageData - Additional TLVs - PXE TLV - Update Length - 0x0081

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PXE TLV - Update length	0x4	

6.2.1.131 FirmwareDevicePackageData - Additional TLVs - PXE TLV - TLV Type - 0x0082

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PXE TLV - TLV Type	0x32	

6.2.1.132 FirmwareDevicePackageData - Additional TLVs - PXE TLV - Update Offset - 0x0083

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PXE TLV - Update offset	0x0	

6.2.1.133 FirmwareDevicePackageData - Additional TLVs - PXE TLV - Data Length - 0x0084

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PXE TLV - Data length	0x1	

6.2.1.134 FirmwareDevicePackageData - Additional TLVs - PXE TLV - Version - 0x0085

Bits	Field Name	Default NVM Value	Description
15:12	Major Version	0x02	
11:8	Minor Version	0x02	
7:0	Build Number	0x01	

6.2.1.135 FirmwareDevicePackageData - Additional TLVs - CIVD TLV - Update Type - 0x0086

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - CIVD TLV - Update type	0x10	

6.2.1.136 FirmwareDevicePackageData - Additional TLVs - CIVD TLV - Update Length - 0x0087

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - CIVD TLV - Update length	0x5	

6.2.1.137 FirmwareDevicePackageData - Additional TLVs - CIVD TLV - TLV Type - 0x0088

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - PLDM TLV - TLV Type	0x132	

6.2.1.138 FirmwareDevicePackageData - Additional TLVs - CIVD TLV - Update Offset - 0x0089

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - CIVD TLV - Update offset	0x0	

6.2.1.139 FirmwareDevicePackageData - Additional TLVs - CIVD TLV - Data Length - 0x008A

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - CIVD TLV - Data length	0x2	

6.2.1.140 FirmwareDevicePackageData - Additional TLVs - CIVD TLV - Version High - 0x008B

Bits	Field Name	Default NVM Value	Description
15:8	Major	0x01	
7:0	Build	0x0B	

6.2.1.141 FirmwareDevicePackageData - Additional TLVs - CIVD TLV - Version Low - 0x008C

Bits	Field Name	Default NVM Value	Description
15:8	Build	0x17	
7:0	Patch	0x00	

6.2.1.142 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Update Type - 0x008D

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Update type	0x10	

6.2.1.143 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Update Length - 0x008E

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - IBA TLV - Update length	0x27	

6.2.1.144 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - TLV Type - 0x008F

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - TLV Type	0x128	

6.2.1.145 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Update Offset - 0x0090

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Update Offset	0x0	

6.2.1.146 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Data Length - 0x0091

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Data Length	0x24	

6.2.1.147 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Dell identifier - 0x0092

Bits	Field Name	Default NVM Value	Description
15:0	OEM identifier	0x2A2	

6.2.1.148 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - CEM Flags - 0x0093

Bits	Field Name	Default NVM Value	Description
15:7	Max Temperature	0x69	Max Temperature reported in Get Temperature Response.
6:2	Reserved	0x0	
1	Support LegacyBootOrder	0x0	If set, the LegacyBootOrder TLV is enabled.
0	Reserved	0x0	

6.2.1.149 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Number of Inventory NVM Entries - 0x0094

Bits	Field Name	Default NVM Value	Description
15:8	Reserved	0x0	Reserved for alignment.
7:0	Number of Entries	0x4	Number of items stored in this Inventory NVM Structure. Each item is formed by three fields: ID (1Byte), Length in bytes (1Byte), Data (variable length).

6.2.1.150 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Media Type ID Length - 0x0095

Bits	Field Name	Default NVM Value	Description
15:8	Length	0x2	Length in bytes of the Media Type item.
7:0	Id	0x4	ID for Network Type item.

6.2.1.151 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Media Type Data - 0x0096

Bits	Field Name	Default NVM Value	Description
15:0	Data	0x0	This parameter is a bitmap that reflects the type of networks/connectivity supported by the device. The entries in bold are used for connectivity and should be used Bit Media types: 0 = BASE-T is supported. 1 = KR is supported. 2 = KX is supported. 3 = KX4 is supported. 4 = SR is supported. 5 = SFP is supported. 6 = SFP+ is supported. 7 = DCA is supported. 8 = FC is supported. 9 = Backplane is supported. 10 = SFF Cage is supported. 11-15 = Not used

6.2.1.152 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Number of Extended Capabilities Entries - 0x0097

Bits	Field Name	Default NVM Value	Description
15:8	Reserved	0x0	Reserved for alignment.
7:0	Number of Entries	0x4	Number of items stored in this Inventory NVM Structure. Each item is formed by three fields: ID (1Byte), Length in bytes (1Byte), Data (variable length).

6.2.1.153 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Capabilities Flags ID Length - 0x0098

Bits	Field Name	Default NVM Value	Description
15:8	Length	0x4	Length in bytes of the Capabilities Flags item.
7:0	Id	0x0	ID for Capabilities Flags item.

6.2.1.154 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Capabilities Flags Data High - 0x0099

Bits	Field Name	Default NVM Value	Description
15:0	Data High	0x0408	The Capabilities Flags bitmap.

6.2.1.155 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Capabilities Flags Data Low - 0x009A

Bits	Field Name	Default NVM Value	Description
15:0	Data Low	0x01FB	Bit Supported Feature 0 = Virtual Addressing 1 = Link Tuning 2 = Remote PHY 3 = Option ROM 4 = EFI 5 = WOL 6 = BMC Pass Through 7 = OS-BMC Pass Through 8 = EEE 9 = On Chip Thermal Sensor 10 = Multiple Temperature Reporting 11 = Reserved 12 = TCP Offload 13 = iSCSI Offload 14 = Reserved 15 = Reserved 16 = Reserved 17 = Reserved 18 = PXE 19 = iSCSI Boot 20 = Reserved 21 = Storage Boot Parameter Provisioning 22 = Reserved 23 = Reserved 24 = NIC Partitioning 25 = SR-IOV 26 = Dell License Manager 27 = IPsec (Not Supported) 28 = MACsec (Not Supported) 29 = Reserved 30 = Reserved 31 = Reserved

6.2.1.156 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - DCB Capabilities ID Length - 0x009B

Bits	Field Name	Default NVM Value	Description
15:8	Length	0x2	Length in bytes of the DCB Capabilities item.
7:0	Id	0x1	ID for Capabilities Flags item.

6.2.1.157 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - DCB Capabilities Data - 0x009C

Bits	Field Name	Default NVM Value	Description
15:0	DCB Capabilities Data	0x000F	The DCB Capabilities.

6.2.1.158 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - E-switch Capabilities ID Length - 0x009D

Bits	Field Name	Default NVM Value	Description
15:8	Length	0x2	Length in bytes of the E-switch Capabilities item
7:0	Id	0x1	ID for E-switch Capabilities item.

6.2.1.159 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - E-switch Capabilities Data - 0x009E

Bits	Field Name	Default NVM Value	Description
15:0	E-switch Capabilities Data	0x0009	The E-switch Capabilities. This value should be null terminated if it is less than 32 bytes.

6.2.1.160 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - NIC Partitioning Capabilities ID Length - 0x009F

Bits	Field Name	Default NVM Value	Description
15:8	Length	0x2	
7:0	Id	0x1	

6.2.1.161 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - NIC Partitioning Capabilities Data - 0x00A0

Bits	Field Name	Default NVM Value	Description
15:0	Data	0x0000	Data for NIC Partitioning Capabilities item.

6.2.1.162 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - Number of Partition Information NVM Entries - 0x00A1

Bits	Field Name	Default NVM Value	Description
15:8	LAN Personality ID	0x0	ID for LAN personality item.
7:0	Number of Inventory NVM Entries	0x1	Number of items stored in this Inventory NVM Structure. Each item is formed by three fields: ID (1Byte), Length in bytes (1Byte), Data (variable length),

6.2.1.163 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality ID Length - 0x00A2

Bits	Field Name	Default NVM Value	Description
15:8	Length	0x1C	
7:0	Id	0x1	

6.2.1.164 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 1+2 - 0x00A3

Bits	Field Name	Default NVM Value	Description
15:0	LAN Personality Data	0x3558	

6.2.1.165 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 3+4 - 0x00A4

Bits	Field Name	Default NVM Value	Description
15:0	LAN Personality Data	0x3035	

6.2.1.166 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 5+6 - 0x00A5

Bits	Field Name	Default NVM Value	Description
15:0	LAN Personality Data	0x3120	

6.2.1.167 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 7+8 - 0x00A6

Bits	Field Name	Default NVM Value	Description
15:0	LAN Personality Data	0x4730	

6.2.1.168 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 9+10 - 0x00A7

Bits	Field Name	Default NVM Value	Description
15:0	LAN Personality Data	0x4562	

6.2.1.169 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 11+12 - 0x00A8

Bits	Field Name	Default NVM Value	Description
15:0	LAN Personality Data	0x430a	

6.2.1.170 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 13+14 - 0x00A9

Bits	Field Name	Default NVM Value	Description
15:0	LAN Personality Data	0x6e6f	

6.2.1.171 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 15+16 - 0x00AA

Bits	Field Name	Default NVM Value	Description
15:0	LAN Personality Data	0x7274	

6.2.1.172 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 17+18 - 0x00AB

Bits	Field Name	Default NVM Value	Description
15:0	LAN Personality Data	0x6c6f	

6.2.1.173 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 19+20 - 0x00AC

Bits	Field Name	Default NVM Value	Description
15:0	LAN Personality Data	0x656c	

6.2.1.174 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 21+22 - 0x00AD

Bits	Field Name	Default NVM Value	Description
15:0	LAN Personality Data	0x2072	

6.2.1.175 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 23+24 - 0x00AE

Bits	Field Name	Default NVM Value	Description
15:0	LAN Personality Data	0x202d	

6.2.1.176 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 25+26 - 0x00AF

Bits	Field Name	Default NVM Value	Description
15:0	LAN Personality Data	0x414c	

6.2.1.177 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 27+28 - 0x00B0

Bits	Field Name	Default NVM Value	Description
15:0	LAN Personality Data	0x004e	

6.2.1.178 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 29+30 - 0x00B1

Bits	Field Name	Default NVM Value	Description
15:0	LAN Personality Data	0x0000	

6.2.1.179 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 30+31 - 0x00B2

Bits	Field Name	Default NVM Value	Description
15:0	LAN Personality Data	0x0000	

6.2.1.180 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 32+33 - 0x00B3

Bits	Field Name	Default NVM Value	Description
15:0	LAN Personality Data	0x0000	

6.2.1.181 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 34+35 - 0x00B4

Bits	Field Name	Default NVM Value	Description
15:0	LAN Personality Data	0x0000	

6.2.1.182 FirmwareDevicePackageData - Additional TLVs - Dell support TLV - LAN Personality Data 36+37 - 0x00B5

Bits	Field Name	Default NVM Value	Description
15:0	LAN Personality Data	0x0000	

6.2.1.183 FirmwareDevicePackageData - Additional TLVs - VPD update - Update Type - 0x00B6

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - VPD update - Update Type	0x11	

6.2.1.184 FirmwareDevicePackageData - Additional TLVs - VPD update - Update Length - 0x00B7

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - VPD update - Update Length	0x08	

6.2.1.185 FirmwareDevicePackageData - Additional TLVs - VPD update - VPD V0 key - 0x00B8

Bits	Field Name	Default NVM Value	Description
15:0	V0 key	0x3056	

6.2.1.186 FirmwareDevicePackageData - Additional TLVs - VPD update - Data length - 0x00B9

Bits	Field Name	Default NVM Value	Description
15:0	Data length	0xC	

6.2.1.187 FirmwareDevicePackageData - Additional TLVs - VPD update - V0 key data0 - 0x00BA

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - VPD update - V0 key data	0x4646	

6.2.1.188 FirmwareDevicePackageData - Additional TLVs - VPD update - V0 key data 1 - 0x00BB

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - VPD update - V0 key data	0x3256	

6.2.1.189 FirmwareDevicePackageData - Additional TLVs - VPD update - V0 key data 2 - 0x00BC

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - VPD update - V0 key data	0x2E30	

6.2.1.190 FirmwareDevicePackageData - Additional TLVs - VPD update - V0 key data 3 - 0x00BD

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - VPD update - V0 key data	0x3030	

6.2.1.191 FirmwareDevicePackageData - Additional TLVs - VPD update - V0 key data 4 - 0x00BE

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - VPD update - V0 key data	0x302E	

6.2.1.192 FirmwareDevicePackageData - Additional TLVs - VPD update - V0 key data 5 - 0x00BF

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - VPD update - V0 key data	0x0030	

6.2.1.193 FirmwareDevicePackageData - Additional TLVs - IBA TLV - Update Type - 0x00C0

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - IBA TLV - Update type	0x10	

6.2.1.194 FirmwareDevicePackageData - Additional TLVs - IBA TLV - Update Length - 0x00C1

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - IBA TLV - Update length	0x4	

6.2.1.195 FirmwareDevicePackageData - Additional TLVs - IBA TLV - TLV Type - 0x00C2

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - IBA TLV - TLV Type	0x33	

6.2.1.196 FirmwareDevicePackageData - Additional TLVs - IBA TLV - Update Offset - 0x00C3

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - IBA TLV - Update offset	0x0	

6.2.1.197 FirmwareDevicePackageData - Additional TLVs - IBA TLV - Data Length - 0x00C4

Bits	Field Name	Default NVM Value	Description
15:0	FirmwareDevicePackageData - Additional TLVs - IBA TLV - Data length	0x1	

6.2.1.198 FirmwareDevicePackageData - Additional TLVs - IBA TLV - Version - 0x00C5

Bits	Field Name	Default NVM Value	Description
15:14	Signature	0x1	
13	Allow PXE Disable	0x0	If set to 0b, PXE is always load regardless the settings in bits 2:0 in Main Setup Options word. If set to 1b, PXE is loaded when bits 2:0 in Main Setup Options are set to 0 (PXE enabled on this port). Valid values are: 0x0 = Ignore PXE Disable 0x1 = Allow PXE Disable
12:6	Reserved	0x0	
5	Reserved	0x0	
4	iSCSI boot	0x0	
3	efi ebd driver	0x1	
2	RPL	0x0	
1	PXE/UNDI Driver	0x1	
0	PXE Base Code	0x1	

6.2.1.199 ComponentImageCount - 0x00C6

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageCount	0x3	

6.2.1.200 ComponentClassification - 0x00C7

Bits	Field Name	Default NVM Value	Description
15:0	ComponentClassification	0xA	Valid values are: 0x0 = Unknown 0x1 = Other 0x2 = Driver 0x3 = Configuration Software 0x4 = Application Software 0x5 = Instrumentation 0x6 = Firmware/BIOS 0x7 = Diagnostic Software 0x8 = Operating System 0x9 = Middleware 0xA = Firmware 0xB = BIOS/FCode 0xC = Support/Service Pack 0xD = Software Bundle

6.2.1.201 ComponentIdentifier - 0x00C8

Bits	Field Name	Default NVM Value	Description
15:0	ComponentIdentifier	0x6	Valid values are: 0x0 = Reserved 0x1 = TLV Extension 0x2 = Link Topology Scratch Pad Area 0x3 = Descriptor Block 0x4 = Reserved 0x5 = OROM 0x6 = NVM 0x7 = Reserved 0x8 = Topology Netlist

6.2.1.202 ComponentComparisonStamp LSB - 0x00C9

Bits	Field Name	Default NVM Value	Description
15:0	ComponentComparisonStamp Low	0x0	

6.2.1.203 ComponentComparisonStamp MSB - 0x00CA

Bits	Field Name	Default NVM Value	Description
15:0	ComponentComparisonStamp High	0x0	

6.2.1.204 ComponentOptions - 0x00CB

Bits	Field Name	Default NVM Value	Description
15:2	Reserved	0x0	
1	Use Component Comparison Stamp	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
0	Force Update	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled

6.2.1.205 RequestedComponentActivationMethod - 0x00CC

Bits	Field Name	Default NVM Value	Description
15:6	Reserved	0x0	
5	AC power cycle	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
4	DC power cycle	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
3	System reboot	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
2	Medium-specific reset	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
1	Self-Contained	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
0	Automatic	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled

6.2.1.206 ComponentLocationOffset LSB - 0x00CD

Bits	Field Name	Default NVM Value	Description
15:0	ComponentLocationOffset	0x0	

6.2.1.207 ComponentLocationOffset MSB - 0x00CE

Bits	Field Name	Default NVM Value	Description
15:0	ComponentLocationOffset	0x0	

6.2.1.208 ComponentSize LSB - 0x00CF

Bits	Field Name	Default NVM Value	Description
15:0	ComponentSize LSB	0x0	Need to fix to contain size of section

6.2.1.209 ComponentSize MSB - 0x00D0

Bits	Field Name	Default NVM Value	Description
15:0	ComponentSize MSB	0x0	

6.2.1.210 ComponentVersionStringTypeAndLength - 0x00D1

Bits	Field Name	Default NVM Value	Description
15:8	ComponentVersionStringLength	0x12	
7:0	ComponentVersionStringType	0x1	Valid values are: 0x0 = Unknown 0x1 = ASCII 0x2 = UTF-8 0x3 = UTF-16 0x4 = UTF-16LE 0x5 = UTF-16BE

6.2.1.211 ComponentVersionString - Dev starter Major - 0x00D2

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Dev starter Major	0x3030	

6.2.1.212 ComponentVersionString - Dev starter Minor - 0x00D3

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Dev starter Minor	0x3030	

6.2.1.213 ComponentVersionString - EETRACK-ID MSB - 0x00D4

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - EETRACK-ID MSB	0x3030	MSB of EETRACK-ID, [17:0] bits of the eetrack-id (in ASCII)

6.2.1.214 ComponentVersionString - EETRACK-ID LSB - 0x00D5

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - EETRACK-ID MSB	0x3030	

6.2.1.215 ComponentVersionString - Dot and Srev Byte7 - 0x00D6

Bits	Field Name	Default NVM Value	Description
15:8	ComponentVersionString - Srev Byte7	0x30	MSB of 8 byte SREV version (in ASCII)
7:0	ComponentVersionString - Dot	0x2E	

6.2.1.216 ComponentVersionString - Srev Bytes 6-5 - 0x00D7

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Srev Bytes 6-5	0x3030	

6.2.1.217 ComponentVersionString - Srev Bytes 4-3 - 0x00D8

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Srev Bytes 4-3	0x3030	

6.2.1.218 ComponentVersionString - Srev Bytes 2-1 - 0x00D9

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Srev Bytes 2-1	0x3030	

6.2.1.219 ComponentVersionString - Srev Byte 0 and null - 0x00DA

Bits	Field Name	Default NVM Value	Description
15:8	ComponentVersionString - null	0x0	
7:0	ComponentVersionString - Srev LSB	0x30	LSB of 8 byte SREV version (in ASCII)

6.2.1.220 ComponentClassification - 0x00DB

Bits	Field Name	Default NVM Value	Description
15:0	ComponentClassification	0xA	Valid values are: 0x0 = Unknown 0x1 = Other 0x2 = Driver 0x3 = Configuration Software 0x4 = Application Software 0x5 = Instrumentation 0x6 = Firmware/BIOS 0x7 = Diagnostic Software 0x8 = Operating System 0x9 = Middleware 0xA = Firmware 0xB = BIOS/FCode 0xC = Support/Service Pack 0xD = Software Bundle

6.2.1.221 ComponentIdentifier - 0x00DC

Bits	Field Name	Default NVM Value	Description
15:0	ComponentIdentifier	0x5	Valid values are: 0x0 = Reserved 0x1 = TLV Extension 0x2 = Link Topology Scratch Pad Area 0x3 = Descriptor Block 0x4 = Reserved 0x5 = OROM 0x6 = NVM 0x7 = Reserved 0x8 = Topology Netlist

6.2.1.222 ComponentComparisonStamp LSB - 0x00DD

Bits	Field Name	Default NVM Value	Description
15:0	ComponentComparisonStamp Low	0x0	

6.2.1.223 ComponentComparisonStamp MSB - 0x00DE

Bits	Field Name	Default NVM Value	Description
15:0	ComponentComparisonStamp High	0x0	

6.2.1.224 ComponentOptions - 0x00DF

Bits	Field Name	Default NVM Value	Description
15:2	Reserved	0x0	
1	Use Component Comparison Stamp	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
0	Force Update	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled

6.2.1.225 RequestedComponentActivationMethod - 0x00E0

Bits	Field Name	Default NVM Value	Description
15:6	Reserved	0x0	
5	AC power cycle	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
4	DC power cycle	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
3	System reboot	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
2	Medium-specific reset	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
1	Self-Contained	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
0	Automatic	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled

6.2.1.226 ComponentLocationOffset LSB - 0x00E1

Bits	Field Name	Default NVM Value	Description
15:0	ComponentLocationOffset	0x0	

6.2.1.227 ComponentLocationOffset MSB - 0x00E2

Bits	Field Name	Default NVM Value	Description
15:0	ComponentLocationOffset	0x0	

6.2.1.228 ComponentSize LSB - 0x00E3

Bits	Field Name	Default NVM Value	Description
15:0	ComponentSize LSB	0x0	Need to fix to contain size of section.

6.2.1.229 ComponentSize MSB - 0x00E4

Bits	Field Name	Default NVM Value	Description
15:0	ComponentSize MSB	0x0	

6.2.1.230 ComponentVersionStringTypeAndLength - 0x00E5

Bits	Field Name	Default NVM Value	Description
15:8	ComponentVersionStringLength	0x12	
7:0	ComponentVersionStringType	0x1	Valid values are: 0x0 = Unknown 0x1 = ASCII 0x2 = UTF-8 0x3 = UTF-16 0x4 = UTF-16LE 0x5 = UTF-16BE

6.2.1.231 ComponentVersionString - CIVD High MSB - 0x00E6

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - CIVD High MSB	0x3030	

6.2.1.232 ComponentVersionString - CIVD High LSB - 0x00E7

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - CIVD High LSB	0x3030	

6.2.1.233 ComponentVersionString - CIVD Low MSB - 0x00E8

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - CIVD Low MSB	0x3030	

6.2.1.234 ComponentVersionString - CIVD Low LSB - 0x00E9

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - CIVD Low LSB	0x3030	

6.2.1.235 ComponentVersionString - Dot and Srev Byte7 - 0x00EA

Bits	Field Name	Default NVM Value	Description
15:8	ComponentVersionString - Srev Byte7	0x30	MSB of 8-byte SREV version (in ASCII).
7:0	ComponentVersionString - Dot	0x2E	

6.2.1.236 ComponentVersionString - Srev Bytes 6-5 - 0x00EB

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Srev Bytes 6-5	0x3030	

6.2.1.237 ComponentVersionString - Srev Bytes 4-3 - 0x00EC

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Srev Bytes 4-3	0x3030	

6.2.1.238 ComponentVersionString - Srev Bytes 2-1 - 0x00ED

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Srev Bytes 2-1	0x3030	

6.2.1.239 ComponentVersionString - Srev Byte 0 and null - 0x00EE

Bits	Field Name	Default NVM Value	Description
15:8	ComponentVersionString - null	0x0	
7:0	ComponentVersionString - Srev LSB	0x30	LSB of 8-byte SREV version (in ASCII).

6.2.1.240 ComponentClassification - 0x00EF

Bits	Field Name	Default NVM Value	Description
15:0	ComponentClassification	0xA	Valid values are: 0x0 = Unknown 0x1 = Other 0x2 = Driver 0x3 = Configuration Software 0x4 = Application Software 0x5 = Instrumentation 0x6 = Firmware/BIOS 0x7 = Diagnostic Software 0x8 = Operating System 0x9 = Middleware 0xA = Firmware 0xB = BIOS/FCode 0xC = Support/Service Pack 0xD = Software Bundle

6.2.1.241 ComponentIdentifier - 0x00F0

Bits	Field Name	Default NVM Value	Description
15:0	ComponentIdentifier	0x8	Valid values are: 0x0 = Reserved 0x1 = TLV Extension 0x2 = Link Topology Scratch Pad Area 0x3 = Descriptor Block 0x4 = Reserved 0x5 = OROM 0x6 = NVM 0x7 = Reserved 0x8 = Topology Netlist

6.2.1.242 ComponentComparisonStamp LSB - 0x00F1

Bits	Field Name	Default NVM Value	Description
15:0	ComponentComparisonStamp Low	0x0	

6.2.1.243 ComponentComparisonStamp MSB - 0x00F2

Bits	Field Name	Default NVM Value	Description
15:0	ComponentComparisonStamp High	0x0	

6.2.1.244 ComponentOptions - 0x00F3

Bits	Field Name	Default NVM Value	Description
15:2	Reserved	0x0	
1	Use Component Comparison Stamp	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
0	Force Update	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled

6.2.1.245 RequestedComponentActivationMethod - 0x00F4

Bits	Field Name	Default NVM Value	Description
15:6	Reserved	0x0	
5	AC power cycle	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
4	DC power cycle	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
3	System reboot	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
2	Medium-specific reset	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
1	Self-Contained	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
0	Automatic	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled

6.2.1.246 ComponentLocationOffset LSB - 0x00F5

Bits	Field Name	Default NVM Value	Description
15:0	ComponentLocationOffset	0x0	

6.2.1.247 ComponentLocationOffset MSB - 0x00F6

Bits	Field Name	Default NVM Value	Description
15:0	ComponentLocationOffset	0x0	

6.2.1.248 ComponentSize LSB - 0x00F7

Bits	Field Name	Default NVM Value	Description
15:0	ComponentSize LSB	0x0	Need to fix to contain size of section.

6.2.1.249 ComponentSize MSB - 0x00F8

Bits	Field Name	Default NVM Value	Description
15:0	ComponentSize MSB	0x0	

6.2.1.250 ComponentVersionStringTypeAndLength - 0x00F9

Bits	Field Name	Default NVM Value	Description
15:8	ComponentVersionStringLength	0x2A	
7:0	ComponentVersionStringType	0x1	Valid values are: 0x0 = Unknown 0x1 = ASCII 0x2 = UTF-8 0x3 = UTF-16 0x4 = UTF-16LE 0x5 = UTF-16BE

6.2.1.251 ComponentVersionString - ReleaseVersion Major Bytes 7-6 - 0x00FA

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - ReleaseVersion Major Bytes 7-6	0x3030	

6.2.1.252 ComponentVersionString - ReleaseVersion Major Bytes 5-4 - 0x00FB

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - ReleaseVersion Major Bytes 5-4	0x3030	

6.2.1.253 ComponentVersionString - ReleaseVersion Major Bytes 3-2 - 0x00FC

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - ReleaseVersion Major Bytes 3-2	0x3030	

6.2.1.254 ComponentVersionString - ReleaseVersion Major Bytes 1-0 - 0x00FD

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - ReleaseVersion Major Bytes 1-0	0x3030	

6.2.1.255 ComponentVersionString - Dot and ReleaseVersion Minor Byte 7 - 0x00FE

Bits	Field Name	Default NVM Value	Description
15:8	ComponentVersionString - Dot and ReleaseVersion Minor Byte 7	0x30	
7:0	ComponentVersionString - Dot	0x2E	

6.2.1.256 ComponentVersionString - ReleaseVersion Minor Bytes 6-5 - 0x00FF

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - ReleaseVersion Minor Bytes 6-5	0x3030	

6.2.1.257 ComponentVersionString - ReleaseVersion Minor Bytes 4-3 - 0x0100

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - ReleaseVersion Minor Bytes 4-3	0x3030	

6.2.1.258 ComponentVersionString - ReleaseVersion Minor Bytes 2-1 - 0x0101

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - ReleaseVersion Minor Bytes 2-1	0x3030	

6.2.1.259 ComponentVersionString - ReleaseVersion Minor Byte 0 and Dot - 0x0102

Bits	Field Name	Default NVM Value	Description
15:8	ComponentVersionString - Dot	0x2E	
7:0	ComponentVersionString - ReleaseVersion Minor Byte 0	0x30	

6.2.1.260 ComponentVersionString - ReleaseVersion Type Bytes 7-6 - 0x0103

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - ReleaseVersion Type Bytes 7-6	0x3030	

6.2.1.261 ComponentVersionString - ReleaseVersion Type Bytes 5-4 - 0x0104

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - ReleaseVersion Type Bytes 5-4	0x3030	

6.2.1.262 ComponentVersionString - ReleaseVersion Type Bytes 3-2 - 0x0105

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - ReleaseVersion Type Bytes 3-2	0x3030	

6.2.1.263 ComponentVersionString - ReleaseVersion Type Bytes 0-1 - 0x0106

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - ReleaseVersion Type Bytes 0-1	0x3030	

6.2.1.264 ComponentVersionString - Dot and Customer Netlist IANA Byte 7 - 0x0107

Bits	Field Name	Default NVM Value	Description
15:8	ComponentVersionString - Customer Netlist IANA Byte 7	0x30	
7:0	ComponentVersionString - Dot	0x2E	

6.2.1.265 ComponentVersionString - Customer Netlist IANA Bytes 6-5 - 0x0108

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Customer Netlist IANA Bytes 6-5	0x3030	

6.2.1.266 ComponentVersionString - Customer Netlist IANA Bytes 4-3 - 0x0109

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Customer Netlist IANA Bytes 4-3	0x3030	

6.2.1.267 ComponentVersionString - Customer Netlist IANA Bytes 2-1 - 0x010A

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Customer Netlist IANA Bytes 2-1	0x3030	

6.2.1.268 ComponentVersionString - Customer Netlist IANA Byte 0 and Dot - 0x010B

Bits	Field Name	Default NVM Value	Description
15:8	ComponentVersionString - Dot	0x2E	
7:0	ComponentVersionString - Customer Netlist IANA Byte 0	0x30	

6.2.1.269 ComponentVersionString - Customer Netlist Version Bytes 3-2 - 0x010C

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Customer Netlist Version Bytes 3-2	0x3030	

6.2.1.270 ComponentVersionString - Customer Netlist Version Bytes 1-0 - 0x010D

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - Customer Netlist Version Bytes 1-0	0x3030	

6.2.1.271 ComponentVersionString - nulls - 0x010E

Bits	Field Name	Default NVM Value	Description
15:0	ComponentVersionString - nulls	0x0	

6.2.1.272 PackageHeaderChecksum - LSB - 0x010F

Bits	Field Name	Default NVM Value	Description
15:0	PackageHeaderChecksum		CRC-32: Start Section -> Word: PLDM Header -> PackageHeaderIdentifier_0 End Section -> Word: PLDM Header -> ComponentVersionString - nulls

6.2.1.273 PackageHeaderChecksum - MSB - 0x0110

Bits	Field Name	Default NVM Value	Description
15:0	PackageHeaderChecksum		

6.3 NVM Content

6.3.1 SPI Descriptor Section Summary Table

Word Offset	Description	Reference
0x0000	SPI flash descriptor	See section 6.3.1.1

6.3.1.1 SPI flash descriptor - 0x0000

Raw data module length: 2048 words

6.3.2 Init Module Section Summary Table

This is the NVM header module which contains pointers to all other 1st level sections. It includes also words which are relative to the whole NVM map.

Word Offset	Description	Reference
0x0000	NVM Control Word 1	See section 6.3.2.1
0x0001	Non-persistent End Pointer	See section 6.3.2.2
0x0002	Last PFA word Pointer	See section 6.3.2.3
0x0003	GFID pointer	See section 6.3.2.4
0x0004 - 0x0006	Reserved	See section 6.3.2.5
0x0007	Auto Generated Pointers Pointer	See section 6.3.2.6
0x0008	Reserved	See section 6.3.2.7
0x0009	EMP Global Module Pointer	See section 6.3.2.8
0x000A	Guarded Zone Pointer	See section 6.3.2.9
0x000B	EMP Image Pointer	See section 6.3.2.10
0x000C - 0x000D	Reserved	See section 6.3.2.11
0x000E	Manageability Module Pointer	See section 6.3.2.12
0x000F	EMP Settings Module Pointer	See section 6.3.2.13
0x0010	SW Compatibility Word 1	See section 6.3.2.14
0x0011	SW Compatibility Word 2	See section 6.3.2.15
0x0012	SW Compatibility Word 3	See section 6.3.2.16
0x0013	SW Compatibility Word 4	See section 6.3.2.17
0x0014	SW Compatibility Word 5	See section 6.3.2.18
0x0015 - 0x0017	Reserved	See section 6.3.2.19
0x0018	Software Reserved Word 1 - Dev Starter Version	See section 6.3.2.20
0x0019	Software Reserved Word 2	See section 6.3.2.21
0x001A	Software Reserved Word 3	See section 6.3.2.22

Word Offset	Description	Reference
0x001B	Software Reserved Word 4 - OEM Product Version Address Block Pointer	See section 6.3.2.23
0x001C + 1*n, n=5...17	Software Reserved Word	See section 6.3.2.24
0x0029	Software Reserved Word 18 - Map Version	See section 6.3.2.25
0x002A	Software Reserved Word 19 - NVM_Image_Version	See section 6.3.2.26
0x002B	Software Reserved Word 20 - NVM Structure Version	See section 6.3.2.27
0x002C	Software Reserved Word 21	See section 6.3.2.28
0x002D	Software Reserved Word 22 - EETRACK ID 1	See section 6.3.2.29
0x002E	Software Reserved Word 23 - EETRACK ID 2	See section 6.3.2.30
0x002F - 0x0033	Reserved	See section 6.3.2.31
0x0034	Software Reserved Word 24 - Original EETRACK ID 1	See section 6.3.2.32
0x0035	Software Reserved Word 25 - Original EETRACK ID 2	See section 6.3.2.33
0x0036 - 0x003A	Reserved	See section 6.3.2.34
0x003B - 0x003C	Reserved	See section 6.3.2.35
0x003D	PHY configuration scripts (DNL) Pointer	See section 6.3.2.36
0x003E - 0x003F	Reserved	See section 6.3.2.37
0x0040	Preserved Field Area pointer	See section 6.3.2.38
0x0041	Reserved	See section 6.3.2.39
0x0042	1st NVM Bank pointer	See section 6.3.2.40
0x0043	NVM Bank Area Size	See section 6.3.2.41
0x0044	1st OROM Bank pointer	See section 6.3.2.42
0x0045	OROM Bank Area Size	See section 6.3.2.43
0x0046	1st TLV extension bank pointer	See section 6.3.2.44
0x0047	TLV Extension Bank Area Size	See section 6.3.2.45
0x0048	EMP SR Settings Pointer	See section 6.3.2.46
0x0049 - 0x004A	Reserved	See section 6.3.2.47
0x004B	Link Topology Scratch Pad Area Pointer	See section 6.3.2.48
0x004C	Link Topology Scratch Pad Area Size	See section 6.3.2.49
0x004D	Configuration Metadata Pointer	See section 6.3.2.50
0x004E - 0x004F	Reserved	See section 6.3.2.51
0x0050	FW Scratch Pad Area Pointer	See section 6.3.2.52
0x0051	FW Scratch Pad Area Size	See section 6.3.2.53
0x0052 - 0x0053	Reserved	See section 6.3.2.54
0x0054	Analog PHY Configuration Module Pointer	See section 6.3.2.55
0x0055	Soft SKUs	See section 6.3.2.56
0x0056	Reserved	See section 6.3.2.57
0x0057	Recovery Firmware Pointer	See section 6.3.2.58
0x0058 - 0x005C	Reserved	See section 6.3.2.59
0x005D	Allow List Pointer	See section 6.3.2.60

Word Offset	Description	Reference
0x005E	Sideband Autoload Pointer	See section 6.3.2.61
0x005F	RDE Dictionaries Pointer	See section 6.3.2.62
0x0060 - 0x0061	Reserved	See section 6.3.2.63
0x0062	Factory Settings Size	See section 6.3.2.64
0x0063	Mailbox Register Autoload Pointer	See section 6.3.2.65
0x0064 - 0x0065	Reserved	See section 6.3.2.66
0x0066	Key Revocation Section Pointer	See section 6.3.2.67
0x0067	ISK Section Pointer	See section 6.3.2.68
0x0068	External Topology Device Image 0 Pointer	See section 6.3.2.69
0x0069	External Topology Device Image 1 Pointer	See section 6.3.2.70
0x006A	External Topology Device Image 2 Pointer	See section 6.3.2.71
0x006B	External Topology Device Image 3 Pointer	See section 6.3.2.72
0x006C	SPDM Certificate slot 0 Pointer	See section 6.3.2.73
0x006D	SPDM Certificate slot 1 Pointer	See section 6.3.2.74
0x006E	SPDM Certificate slot 2 Pointer	See section 6.3.2.75
0x006F	SPDM Certificate slot 3 Pointer	See section 6.3.2.76
0x0070	MAC Reset 0 Registers Autoload Pointer	See section 6.3.2.77
0x0071	MAC Reset 1 Registers Autoload Pointer	See section 6.3.2.78
0x0072	Master Reset 0 Registers Autoload Pointer	See section 6.3.2.79
0x0073	Master Reset 1 Registers Autoload Pointer	See section 6.3.2.80
0x0074	OEM Port Config 0 Pointer	See section 6.3.2.81
0x0075	OEM Port Config 1 Pointer	See section 6.3.2.82
0x0076	NVM Control Word 2	See section 6.3.2.83
0x0077	NVM Hash Module Pointer	See section 6.3.2.84
0x0078 + 1*n, n=19...150	Spare NVM Header Words	See section 6.3.2.85
0x00FC	Legacy NVM Control Word 1	See section 6.3.2.86
0x00FD	Legacy NVM Control Word 2	See section 6.3.2.87
0x00FE	Legacy NVM Control Word 3	See section 6.3.2.88
0x00FF	Legacy NVM Control Word 4	See section 6.3.2.89

6.3.2.1 NVM Control Word 1 - 0x0000

Bits	Field Name	Default NVM Value	Description
15:8	minSREV	0x0	Minimum Security Revision. In a new NVM image, this field shall be generated to match the security revision of the image itself. In the Shadow RAM, this value indicates to the ROM which SREV values are allowed to boot. If the SREV is stored in NVM, then the Mini-Loader image must have a SREV value greater than or equal to the minSREV value found here in order for the image to boot. This value is also consulted during NVM update to determine whether an image is accepted or rejected. When the FW performs a Shadow RAM dump or any other NVM update flow which affects this word, this field will be preserved, and not overwritten by the one in the NVM bank itself. The only way this field changes is via the SREV opt-in flow. Valid values: 0x0 through 0xFE. A value of 0xFF is treated as if it is 0x0. If the SREV is stored in fuses, then this field is ignored by the ROM, EML and FW.
7:6	Bank Validity	0x1	The Bank Validity (Signature) field indicates to the device that there is a valid NVM present. If it is not 01b, the other bits in this word are ignored, no further NVM read is performed, and the default values are used for the configuration space IDs. Valid values are: 0x0 = Value 0x0 0x1 = Shadow Ram Bank Valid 0x2 = Value 0x2 0x3 = Value 0x3
5	Valid NVM Bank Index	0x0	First Bank is located at offset 128KB (pointed by 1st NVM bank pointer). Second Bank is located at 128K + NVM Bank Area Size * 4K Valid values are: 0x0 = First Bank 0x1 = Second Bank
4	Valid TLV Extension Bank Index	0x0	First Bank is located at offset 128KB (pointed by 1st NVM bank pointer). Second Bank is located at 128K + NVM Bank Area Size * 4K Valid values are: 0x0 = First Bank 0x1 = Second Bank
3	Valid OROM Bank Index	0x0	First Bank is located at offset 128KB (pointed by 1st NVM bank pointer). Second Bank is located at 128K + NVM Bank Area Size * 4K Valid values are: 0x0 = First Bank 0x1 = Second Bank
2:0	Reserved	0x0	Place holder for: Number of LPKs =value+1

6.3.2.2 Non-persistent End Pointer - 0x0001

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units. Only the 4KB sector unit is supported for this pointer. Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	Non-persistent End Pointer	0x7FFF	Pointer to the end of the non-persistent shadow RAM area

6.3.2.3 Last PFA word Pointer - 0x0002

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units. Only the 4KB sector unit is supported for this pointer. Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	Last PFA word Pointer	0x0	Pointer to the end of the PFA excluding the PCI ALT module.

6.3.2.4 GFID pointer - 0x0003

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	PCIR GFID Module Pointer	0x0	

Field PCIR GFID Module Pointer points to GFID Module Section. For GFID Module inner structure See [section 6.3.48](#)

6.3.2.5 Reserved - (0x0004 - 0x0006)

Bits	Field Name	Default NVM Value	Description
15:0	RESERVED	0xFFFF	

6.3.2.6 Auto Generated Pointers Pointer - 0x0007

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	Auto Generated Pointers Module Pointer	0x0	

Field Auto Generated Pointers Module Pointer points to Auto Generated Pointers Module Section. For Auto Generated Pointers Module inner structure See [section 6.3.52](#)

6.3.2.7 Reserved - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	RESERVED	0xFFFF	

6.3.2.8 EMP Global Module Pointer - 0x0009

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	EMP Global Module Pointer	0x0	

Field EMP Global Module Pointer points to EMP Global Module Section. For EMP Global Module inner structure See [section 6.3.87](#)

6.3.2.9 Guarded Zone Pointer - 0x000A

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	Guarded Zone Pointer	0x0	

Field Guarded Zone Pointer points to Guarded Zone Section. For Guarded Zone inner structure See [section 6.3.123](#)

6.3.2.10 EMP Image Pointer - 0x000B

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	EMP Image Pointer	0x7FFF	

Field EMP Image Pointer points to EMP Image Section. For EMP Image inner structure See [section 6.3.93](#)

6.3.2.11 Reserved - (0x000C - 0x000D)

Bits	Field Name	Default NVM Value	Description
15:0	RESERVED	0xFFFF	

6.3.2.12 Manageability Module Pointer - 0x000E

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	Manageability Configuration Module Pointer	0x0	

Field Manageability Configuration Module Pointer points to Manageability Module Header Section. For Manageability Module Header inner structure See [section 6.3.49](#)

6.3.2.13 EMP Settings Module Pointer - 0x000F

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	EMP Module Pointer	0x0	

Field EMP Module Pointer points to EMP Settings Module Header Section. For EMP Settings Module Header inner structure See [section 6.3.88](#)

6.3.2.14 SW Compatibility Word 1 - 0x0010

Five words in the NVM image are reserved for compatibility information. New bits within these fields are defined as the need arises for determining software compatibility between various hardware revisions.

Bits	Field Name	Default NVM Value	Description
15:12	Reserved	0x0	Reserved
11	LOM	0x0	Indicates whether the NVM attached to LAN silicon contains dedicated module for option ROM. Used by option ROM update applications. 0b = NIC (Attached flash contains module for option ROM). 1b = LOM (Attached flash has no module for option ROM).
10	Server	0x1	Legacy, not currently used. 0b = Client 1b = Server
9	Reserved	0x0	Reserved

Bits	Field Name	Default NVM Value	Description
8	OEM/Retail	0x0	Legacy, not currently used. 0b = Retail 1b = OEM
7:0	Reserved	0x00	Reserved

6.3.2.15 SW Compatibility Word 2 - 0x0011

Five words in the NVM image are reserved for compatibility information. New bits within these fields are defined as the need arises for determining software compatibility between various hardware revisions.

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved

6.3.2.16 SW Compatibility Word 3 - 0x0012

Five words in the NVM image are reserved for compatibility information. New bits within these fields are defined as the need arises for determining software compatibility between various hardware revisions.

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved

6.3.2.17 SW Compatibility Word 4 - 0x0013

Five words in the NVM image are reserved for compatibility information. New bits within these fields are defined as the need arises for determining software compatibility between various hardware revisions.

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved

6.3.2.18 SW Compatibility Word 5 - 0x0014

Five words in the NVM image are reserved for compatibility information. New bits within these fields are defined as the need arises for determining software compatibility between various hardware revisions.

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved

6.3.2.19 Reserved (0x0015 - 0x0017)

Bits	Field Name	Default NVM Value	Description
15:0	RESERVED	0xFFFF	

6.3.2.20 Software Reserved Word 1 - Dev Starter Version - 0x0018

Dev_Starter map version used to produce this image.

This word shall be filled manually.

Used for upgrade downgrade algorithm.

Bits	Field Name	Default NVM Value	Description
15:12	Major	0x0	NVM major version - Customer Set
11:8	Decimal Point	0x0	Decimal point, used by automatic NVM reading tools. Must be always set to 0x0.
7:0	Minor	0x0	NVM minor version - increased upon each release to customer

6.3.2.21 Software Reserved Word 2 - 0x0019

Bits	Field Name	Default NVM Value	Description
15:2	Reserved	0xFF	Reserved
1	WoL Control Port 1	0x1	Wake On LAN feature for port 1. 1 - Disabled or not supported 0 - Supported and enabled Valid values are: 0x0 = Enabled - Supported and enabled 0x1 = Disabled - Disabled or not supported
0	WoL Control Port 0	0x1	Wake On LAN feature for port 0. 1 - Disabled or not supported 0 - Supported and enabled Valid values are: 0x0 = Enabled - Supported and enabled 0x1 = Disabled - Disabled or not supported

6.3.2.22 Software Reserved Word 3 - 0x001A

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved

6.3.2.23 Software Reserved Word 4 - OEM Product Version Address Block Pointer - 0x001B

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	OEM Product Version Address Block Pointer	0x0	

6.3.2.24 Software Reserved Word[n] (0x001C + 1*n, n=5...17)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	Reserved

6.3.2.25 Software Reserved Word 18 - Map Version - 0x0029

Automatically generated by the tool.

Bits	Field Name	Default NVM Value	Description
15:0	Map Version		

6.3.2.26 Software Reserved Word 19 - NVM_Image_Version - 0x002A

Automatically generated by the tool.

Bits	Field Name	Default NVM Value	Description
15:0	NVM Image Version		

6.3.2.27 Software Reserved Word 20 - NVM Structure Version - 0x002B

Bits	Field Name	Default NVM Value	Description
15:12	Major	0x0	NVM major version
11:8	Decimal Point	0x0	Decimal point, used by automatic NVM reading tools. Must be always set to 0x0.
7:0	Minor	0x0	NVM Structure minor version

6.3.2.28 Software Reserved Word 21 - 0x002C

This word is for platform/NIC/LOM specific settings.

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x3FFF	Reserved

6.3.2.29 Software Reserved Word 22 - EETRACK ID 1 - 0x002D

This word is for the first word of the eTrack_ID number written by EEPROM Manager Tool.

Bits	Field Name	Default NVM Value	Description
15:0	eTrack_ID Word 1	0xFFFF	EEPROM Manager Tool writes a unique 32-bit eTrack_ID number in two sequential NVM words. The eTrack_ID is written when EEPROM Manager Tool creates an image on the Intel network. The eTrack_ID DB tracks NVM images back to a specific SCM build.

6.3.2.30 Software Reserved Word 23 - EETRACK ID 2 - 0x002E

This word is for the second word of the eTrack_ID number written by EEPROM Manager Tool.

Bits	Field Name	Default NVM Value	Description
15:0	eTrack_ID Word 2		

6.3.2.31 Reserved (0x002F - 0x0033)

Bits	Field Name	Default NVM Value	Description
15:0	RESERVED	0xFFFF	

6.3.2.32 Software Reserved Word 24 - Original EETRACK ID 1 - 0x0034

Bits	Field Name	Default NVM Value	Description
15:0	Original EETRACK ID 1	0x0000	

6.3.2.33 Software Reserved Word 25 - Original EETRACK ID 2 - 0x0035

Bits	Field Name	Default NVM Value	Description
15:0	Original EETRACK ID 2	0x0000	

6.3.2.34 Reserved (0x0036 - 0x003A)

Bits	Field Name	Default NVM Value	Description
15:0	RESERVED	0xFFFF	

6.3.2.35 Reserved (0x003B - 0x003C)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	

6.3.2.36 PHY configuration scripts (DNL) Pointer - 0x003D

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	DNL Pointer	0x7FFF	

Field DNL Pointer points to DL Scripts Section. For DL Scripts inner structure See [section 6.3.91](#)

6.3.2.37 Reserved (0x003E - 0x003F)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	

6.3.2.38 Preserved Field Area pointer - 0x0040

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	PFA pointer	0x7FFF	

Field PFA pointer points to PFA Header Section. For PFA Header inner structure See [section 6.3.3](#)

6.3.2.39 Reserved - 0x0041

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	

6.3.2.40 1st NVM Bank pointer - 0x0042

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	1st NVM Bank pointer	0x7FFF	

Field 1st NVM Bank pointer points to NVM Image CSS Header Section. For NVM Image CSS Header inner structure See [section 6.3.56](#)

6.3.2.41 NVM Bank Area Size - 0x0043

Bits	Field Name	Default NVM Value	Description
15:12	RESERVED	0x00	
11:0	NVM Bank Area Size	0x41A	Size expressed in 4 KB sector units.

6.3.2.42 1st OROM Bank pointer - 0x0044

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units
14:0	1st OROM Bank pointer	0x0	Pointer to first OROM section (not swapped when section changes) Note: This value is fixed to 4224 KB - need to be manually changed if location of section changes.

Field 1st OROM Bank pointer points to OROM Section. For OROM inner structure See [section 6.3.101](#)

6.3.2.43 OROM Bank Area Size - 0x0045

Bits	Field Name	Default NVM Value	Description
15:12	RESERVED	0x00	
11:0	OROM Bank Area Size	0x7D	Size expressed in 4 KB sector units.

6.3.2.44 1st TLV extension bank pointer - 0x0046

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	1st TLV extension bank pointer	0x0	Pointer to first TLV extension section (not swapped when section changes) Note: This value is fixed to 4224 KB - need to be manually changed if location of section changes.

Field 1st TLV extension bank pointer points to Link Topology Netlist Section. For Link Topology Netlist inner structure See [section 6.3.110](#)

6.3.2.45 TLV Extension Bank Area Size - 0x0047

Bits	Field Name	Default NVM Value	Description
15:12	RESERVED	0x00	
11:0	TLV Extension Bank Area Size	0x7	Size expressed in 4 KB sector units.

6.3.2.46 EMP SR Settings Pointer - 0x0048

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	EMP SR Settings Pointer	0x7FFF	

Field EMP SR Settings Pointer points to EMP SR Settings Module Header Section. For EMP SR Settings Module Header inner structure See [section 6.3.47](#)

6.3.2.47 Reserved (0x0049 - 0x004A)

Bits	Field Name	Default NVM Value	Description
15:0	RESERVED	0xFFFF	

6.3.2.48 Link Topology Scratch Pad Area Pointer - 0x004B

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	Link Topology Scratch Pad Area Pointer	0x7FFF	

Field Link Topology Scratch Pad Area Pointer points to Link Topology Scratch Pad Area Section. For Link Topology Scratch Pad Area inner structure See [section 6.3.119](#)

6.3.2.49 Link Topology Scratch Pad Area Size - 0x004C

Bits	Field Name	Default NVM Value	Description
15:12	RESERVED	0x00	
11:0	Link Topology Scratch Pad Area Size	0x5	Size expressed in 4 KB sector units.

6.3.2.50 Configuration Metadata Pointer - 0x004D

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	Configuration Metadata Pointer	0x7FFF	

Field Configuration Metadata Pointer points to Configuration Metadata Section. For Configuration Metadata inner structure See [section 6.3.90](#)

6.3.2.51 Reserved (0x004E - 0x004F)

Bits	Field Name	Default NVM Value	Description
15:0	RESERVED	0xFFFF	

6.3.2.52 FW Scratch Pad Area Pointer - 0x0050

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	FW Scratch Pad Area Pointer	0x7FFF	

Field FW Scratch Pad Area Pointer points to FW Scratch Pad Area Section. For FW Scratch Pad Area inner structure See [section 6.3.120](#)

6.3.2.53 FW Scratch Pad Area Size - 0x0051

Bits	Field Name	Default NVM Value	Description
15:10	RESERVED	0x00	
9:0	FW Scratch Pad Area Size	0x40	Size expressed in 4 KB sector units.

6.3.2.54 Reserved (0x0052 - 0x0053)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	

6.3.2.55 Analog PHY Configuration Module Pointer - 0x0054

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	Internal PHY Configuration Module Pointer	0x7FFF	

Field Internal PHY Configuration Module Pointer points to Analog PHY Configuration Section. For Analog PHY Configuration inner structure See [section 6.3.92](#)

6.3.2.56 Soft SKUs - 0x0055

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	Reserved

6.3.2.57 Reserved - 0x0056

Bits	Field Name	Default NVM Value	Description
15:0	RESERVED	0x0	

6.3.2.58 Recovery Firmware Pointer - 0x0057

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	Recovery Firmware Pointer	0x0	

Field Recovery Firmware Pointer points to Recovery FW CSS Header Section. For Recovery FW CSS Header inner structure See [section 6.3.83](#)

6.3.2.59 Reserved (0x0058 -0x005C)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	

6.3.2.60 Allow List Pointer - 0x005D

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units
14:0	Allow List Pointer	0x7FFF	

Field Allow List Pointer points to Allow list Section. For Allow list inner structure See [section 6.3.89](#)

6.3.2.61 Sideband Autoload Pointer - 0x005E

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	Sideband Autoload Pointer	0x7FFF	

Field Sideband Autoload Pointer points to Sideband Bus Autoload Section. For Sideband Bus Autoload inner structure See [section 6.3.46](#)

6.3.2.62 RDE Dictionaries Pointer - 0x005F

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	RDE Dictionaries Pointer	0x0	Pointer to RDE Dictionaries

Field RDE Dictionaries Pointer points to RDE Dictionaries Section. For RDE Dictionaries inner structure See [section 6.3.94](#)

6.3.2.63 Reserved (0x0060 - 0x0061)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	

6.3.2.64 Factory Settings Size - 0x0062

Bits	Field Name	Default NVM Value	Description
15:12	RESERVED	0x00	
11:0	Factory Settings Size	0xD	Size expressed in 4 KB sector units.

6.3.2.65 Mailbox Register Autoload Pointer - 0x0063

Bits	Field Name	Default NVM Value	Description
15:0	Mailbox Register Autoload Pointer	0x0	

6.3.2.66 Reserved (0x0064 - 0x0065)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	

6.3.2.67 Key Revocation Section Pointer - 0x0066

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	Key Revocation Section Pointer	0x0	

Field Key Revocation Section Pointer points to Signature RoT Key Revocation Module CSS Header Section. For Signature RoT Key Revocation Module CSS Header inner structure See [section 6.3.79](#)

6.3.2.68 ISK Section Pointer - 0x0067

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	ISK Section Pointer	0x7FFF	

Field ISK Section Pointer points to NVM ISK Object CSS Header Section. For NVM ISK Object CSS Header inner structure See [section 6.3.75](#)

6.3.2.69 External Topology Device Image 0 Pointer - 0x0068

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	External Topology Device Image 0 Pointer	0x7FFF	

Field External Topology Device Image 0 Pointer points to External Topology Device Image 0 Section. For External Topology Device Image 0 inner structure See [section 6.3.95](#)

6.3.2.70 External Topology Device Image 1 Pointer - 0x0069

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	External Topology Device Image 1 Pointer	0x7FFF	

Field External Topology Device Image 1 Pointer points to External Topology Device Image 1 Section.
For External Topology Device Image 1 inner structure See [section 6.3.96](#)

6.3.2.71 External Topology Device Image 2 Pointer - 0x006A

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	External Topology Device Image 2 Pointer	0x7FFF	

Field External Topology Device Image 2 Pointer points to External Topology Device Image 2 Section.
For External Topology Device Image 2 inner structure See [section 6.3.97](#)

6.3.2.72 External Topology Device Image 3 Pointer - 0x006B

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	External Topology Device Image 3 Pointer	0x7FFF	

Field External Topology Device Image 3 Pointer points to External Topology Device Image 3 Section.
For External Topology Device Image 3 inner structure See [section 6.3.98](#)

6.3.2.73 SPDM Certificate slot 0 Pointer - 0x006C

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	SPDM Certificate slot0 Pointer	0x7FFF	

6.3.2.74 SPDM Certificate slot 1 Pointer - 0x006D

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	SPDM Certificate slot 1 Pointer	0x7FFF	

6.3.2.75 SPDM Certificate slot 2 Pointer - 0x006E

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	SPDM Certificate slot 2 Pointer	0x7FFF	

6.3.2.76 SPDM Certificate slot 3 Pointer - 0x006F

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	SPDM Certificate slot 3 Pointer	0x0	

6.3.2.77 MAC Reset 0 Registers Autoload Pointer - 0x0070

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	MAC Reset Registers Autoload Module Pointer	0x7FFF	

Field MAC Reset Registers Autoload Module Pointer points to MAC Reset 0 Registers Autoload Module Section. For MAC Reset 0 Registers Autoload Module inner structure See [section 6.3.53](#)

6.3.2.78 MAC Reset 1 Registers Autoload Pointer - 0x0071

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	MAC Reset Registers Autoload Module Pointer	0x7FFF	

Field MAC Reset Registers Autoload Module Pointer points to MAC Reset 1 Registers Autoload Module Section. For MAC Reset 1 Registers Autoload Module inner structure See [section 6.3.53](#)

6.3.2.79 Master Reset 0 Registers Autoload Pointer - 0x0072

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	Master Reset Registers Autoload Module Pointer	0x7FFF	

Field Master Reset Registers Autoload Module Pointer points to Master Reset 0 Registers Autoload Module Section. For Master Reset 0 Registers Autoload Module inner structure See [section 6.3.54](#)

6.3.2.80 Master Reset 1 Registers Autoload Pointer - 0x0073

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	Master Reset Registers Autoload Module Pointer	0x7FFF	

Field Master Reset Registers Autoload Module Pointer points to Master Reset 1 Registers Autoload Module Section. For Master Reset 1 Registers Autoload Module inner structure See [section 6.3.54](#)

6.3.2.81 OEM Port Config 0 Pointer - 0x0074

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	OEM Port Config Pointer	0x7FFF	

Field OEM Port Config Pointer points to OEM Port Config 0 Section. For OEM Port Config 0 inner structure See [section 6.3.55](#)

6.3.2.82 OEM Port Config 1 Pointer - 0x0075

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	OEM Port Config Pointer	0x7FFF	

Field OEM Port Config Pointer points to OEM Port Config 1 Section. For OEM Port Config 1 inner structure See [section 6.3.55](#)

6.3.2.83 NVM Control Word 2 - 0x0076

Bits	Field Name	Default NVM Value	Description
15:1	RESERVED	0x0	
0	prevent_failover	0x0	When this bit is set, failover during boot shall be prevented. This means that failover will never be done (also recovery will never be reached). Always the active bank only will be attempted.

6.3.2.84 NVM Hash Module Pointer - 0x0077

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	NVM Hash Module Pointer	0x7FFF	

Field NVM Hash Module Pointer points to NVM Hash Module Section. For NVM Hash Module inner structure See [section 6.3.99](#)

6.3.2.85 Spare NVM Header Words[n] (0x0078+ 1*n, n=18...150)

Bits	Field Name	Default NVM Value	Description
15:0	RESERVED	0xFFFF	

6.3.2.86 Legacy NVM Control Word 1 - 0x00FC

Bits	Field Name	Default NVM Value	Description
15	Reserved (Legacy - SGVL BDX)	0x0	Use slow cycle time for Auto Load of PCI Analog Like MAC and LAN Autoload cycle time
14	Reserved (Legacy - NC-SI HW ARB Disable)	0x1	NC-SI Hardware Arbitration Disable 1b = NCSI_ARB_IN and NCSI_ARB_OUT pads are not used. NCSI_ARB_IN is pulled up internally to provide stable input. 0b = NCSI_ARB_IN and NCSI_ARB_OUT pads are used. Valid values are: 0x0 = Enabled 0x1 = Disabled
13:12	Reserved		Reserved.
11	Reserved (Legacy - SGVL BDX - Fast SPI)	0x0	Reserved Valid values are: 0x0 = Disabled - 12.5Mhz SPI 0x1 = Enabled - 25Mhz SPI
10:8	Reserved (Legacy - NVM Size)	0x5	These bits indicate the Flash's actual size. Mapped to FLA.FL_SISE (see field definition in the FLA register section). Valid values are: 0x4 = 1 MB 0x5 = 2 MB 0x6 = 4 MB
7:6	Reserved (Legacy - Signature)	0x1	The Signature field indicates to the device that there is a valid NVM present. If the Signature field is not 01b, the other bits in this word are ignored, no further NVM read is performed, and the default values are used for the configuration space IDs. Valid values are: 0x0 = NVM not present 0 0x1 = NVM present 0x2 = NVM not present 2 0x3 = NVM not present 3

Bits	Field Name	Default NVM Value	Description
5	Reserved (Legacy - VPD Write Enable)	0x1	When set to 1b, the host can modify the entire VPD area via the Flash Update host I/F command. When set to 0b, any host attempt to modify the VPD area via the Flash Update host I/F command is completed with an error. Note: RW-VPD area can be modified from the config space register set regardless of the setting of this field. Valid values are: 0x0 = Disabled 0x1 = Enabled
4:0	Reserved		Reserved.

6.3.2.87 Legacy NVM Control Word 2 - 0x00FD

Bits	Field Name	Default NVM Value	Description
15	PCI LAN Function Sel -- Swap	0x0	When both LAN ports are enabled and the LAN Function Sel equals 0b, LAN 0 is routed to PCI function 0 and LAN 1 is routed to PCI function 1. If the LAN Function Sel equals 1b, LAN 0 is routed to PCI function 1 and LAN 1 is routed to PCI function 0. This bit is loaded from NVM.
14	Reserved_for_phy_1	0x0	
13	Reserved_for_phy_0	0x0	
12	Reserved	0x0	Reserved
11	Reserved (Legacy - PCI SDP Function OFF Enable)	0x0	PCIe Function Off via SDP Pins Enable. 0b = Legacy mode (default), SDPn_1 pins does not control PCIe functions off. 1b = Two SDP pins are used in conjunction by strapping (sampled by PE_RST_N) to disable the two PCIe functions altogether. Note: If MNG is present, MC-to-LAN paths are not disabled. PCIe Function Off pins can also work independently, where each pin controls the corresponding PCIe function. However independent mode is not supported in this device.
10	Reserved (Legacy - PCI Device OFF Enable)	0x0	This bit is relevant only when the device is disabled via strapping during PE_RST_N both LANn_DIS_N pins to 0b at once. 0b = Legacy mode (default). Though the device is disabled, the digital I/O pins are not moved to an electrical off state. 1b = Enable device electrical off. When the device is disabled, the digital I/O pins are put at High-Z For example, electrical off state where pull-ups/downs are at their defined values.
9	Reserved (Legacy - LAN PCI Disable Select)	0x0	Selects which port is disable
8	Reserved (Legacy - LAN PCI Disable)	0x0	When set one PCI LAN port is disabled
7:4	Reserved	0x0	Reserved
3	Reserved (Legacy - Deadlock Timeout Enable)	0x1	If set, a device that was granted access to the NVM that does not toggle the interface for 2 seconds will have the grant revoked. Valid values are: 0x0 = Timeout Disabled 0x1 = Timeout Enabled
2	Reserved	0x0	
1	Core Clocks Gate Disable	0x0	During nominal operation this bit should be zero enabling core clock gating in low power state. When set disables the gating of the core clock in low power state. Valid values are: 0x0 = Enable - Enables gating of the core clock in low power state 0x1 = Disable - Disables gating of the core clock in low power state
0	Reserved	0x0	Reserved

6.3.2.88 Legacy NVM Control Word 3 - 0x00FE

Bits	Field Name	Default NVM Value	Description
15	WD_FWRST_EN	0x0	Enable FW Reset on Watchdog timeout. Defines the default value of FWRSTCTR.WD_FWRST_EN bit. Should be set for normal operation.
14	ECC_FWRST_EN	0x0	Enable FW Reset on ECC error. Defines the default value of FWRSTCTR.ECC_FWRST_EN bit. Should be set for normal operation.
13	Reserved	0x0	Reserved
12	Reserved (Legacy - MAC2PHY_PTRVLD)	0x0	Enable to manually define the validity of the PHY-IMAGEPointer signal. 0b: PHY-Image-Pointer is Not-Valid. 1b: PHY-Image-Pointer is Valid This signal is relevant only when EN-MAC2PHY-PTRVLD = "1" Valid values are: 0x0 = Invalid PHY-Image-Pointer is Not-Valid 0x1 = Valid PHY-Image-Pointer is Valid
11	Reserved (Legacy - EN_MAC2PHY_PTRVLD)	0x0	Enable to manually control the MAC signal to the PHY, which indicates when the PHY can read the PHY-Image-Pointer from the NVM 0b: HW-Control 1b: Manual control Valid values are: 0x0 = HW-Control 0x1 = Manual Control
10:9	Reserved		Reserved
8	Reserved (Legacy - D10GMP Port 1)	0x0	Disable 10 GbE in LPLU for LAN Port 1. When set, LAN port 1 never advertises 10 GbE speed capability when in LPLU state (D3/Dr).
7	Reserved (Legacy - Enable FW LPLU)	0x0	Enable FW LPLU
6	Reserved (Legacy - D10GMP Port 0)	0x0	Disable 10 GbE in LPLU for LAN Port 0. When set, LAN port 0 never advertises 10 GbE speed capability when in LPLU state (D3/Dr).
5	Reserved	0x0	Reserved
4	Reserved	0x0	Reserved
3	Enable LPLU	0x0	Enable the Low Power Link Up Feature. When set, enables a decrease in link speed of the port defined to stay awake in non-D0a states when power policy and power management states dictate it. In order to allow for firmware to control the LPLU flow, this bit must be always disabled
2	Keep_PHY_Link_Up_En	0x1	Enables No PHY Link Down when the MC indicates that the PHY should be kept on. When asserted, this bit prevents changes in power management state to be reflected to the PHYs according to the MMNGC.MNG_VETO bit value. When cleared, the MMNGC.MNG_VETO bit is meaningless.
1	APM Enable Port 1	0x0	Initial value of advanced power management wake up enable in the General Receive Control register (GRC.APME). Mapped to GRC.APME of port 1.
0	APM Enable Port 0	0x0	Initial value of advanced power management wake up enable in the General Receive Control register (GRC.APME). Mapped to GRC.APME of port 0.

6.3.2.89 Legacy NVM Control Word 4 - 0x00FF

Bits	Field Name	Default NVM Value	Description
15	Reserved (Legacy - GPIO selector 7)	0x0	
14	Reserved (Legacy - GPIO selector 6)	0x0	
13	Reserved (Legacy - GPIO selector 5)	0x0	
12	Reserved (Legacy - GPIO selector 4)	0x0	
11	Reserved (Legacy - GPIO selector 3)	0x1	
10	Reserved (Legacy - GPIO selector 2)	0x1	
9	Reserved (Legacy - GPIO selector 1)	0x1	
8	Reserved (Legacy - GPIO selector 0)	0x1	
7	Reserved	0x0	
6:4	Reserved (Legacy - Device Number)	0x0	
3:1	Reserved	0x0	
0	Reserved (Legacy - BMC_CSR_WR_EN)	0x0	

6.3.3 PFA Header Section Summary Table

Preserved Fields Area Header

Word Offset	Description	Reference
0x0000	PFA Length	See section 6.3.3.1

6.3.3.1 PFA Length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	PFA Length	0x2F00	

6.3.4 PFA Features Module Section Summary Table

Word Offset	Description	Reference
0x0101	Sub Module Type - Features	See section 6.3.4.1

6.3.4.1 Sub Module Type - Features - 0x0101

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x49	Valid values are: 0x49 = Feature Configuration

6.3.5 Feature Configuration Padding module Section Summary Table

Preserved Fields Area Header

Word Offset	Description	Reference
0x0000	Sub Module Type padding	See section 6.3.5.1
0x0001	Length	See section 6.3.5.2
0x0002	Padding	See section 6.3.5.3

6.3.5.1 Sub Module Type padding - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0xffff	Valid values are: 0xffff = Padding Module

6.3.5.2 Length - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit - 2 First Section -> Word: Feature Configuration Padding module -> Sub Module Type padding Last Section -> Word: Feature Configuration Padding module -> Padding

6.3.5.3 Padding - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	[New Field]	0xFFFF	

6.3.6 PFA Immediate Values Module Section Summary Table

Preserved Fields Area Header

Word Offset	Description	Reference
0x0901	Sub Module Type - immediate	See section 6.3.6.1

6.3.6.1 Sub Module Type - immediate - 0x0901

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x4E	Valid values are: 0x4E = Immediate Configuration

6.3.7 Immediate Fields Padding module Section Summary Table

Preserved Fields Area Header

Word Offset	Description	Reference
0x0000	Sub Module Type padding	See section 6.3.7.1
0x0001	Length	See section 6.3.7.2
0x0002	Padding	See section 6.3.7.3

6.3.7.1 Sub Module Type padding - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0xffff	Valid values are: 0xffff = Padding Module

6.3.7.2 Length - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit - 2 First Section -> Word: Immediate Fields Padding module -> Sub Module Type padding Last Section -> Word: Immediate Fields Padding module -> Padding

6.3.7.3 Padding - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	[New Field]	0xFFFF	

6.3.8 PFA VPD Module Section Summary Table

VPD area - Vital Product Data loaded by OEM. It contains RO and RW info about the NIC/LOM.

Word Offset	Description	Reference
0x0F01	Sub Module Type - VPD	See section 6.3.8.1
0x0F02	Length	See section 6.3.8.2
0x0F03	VPD Data	See section 6.3.8.3

6.3.8.1 Sub Module Type - VPD - 0x0F01

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x2F	Valid values are: 0x2F = VPD Module

6.3.8.2 Length - 0x0F02

Bits	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit - 2 First Section -> Word: PFA VPD Module -> Sub Module Type - VPD Last Section -> Word: PFA VPD Module -> VPD Data

6.3.8.3 VPD Data - 0x0F03

Raw data module length: 512 words

6.3.9 PFA MNG Filter Section Summary Table

Preserved Fields Area Header

Word Offset	Description	Reference
0x0000	Sub Module Type - MNG filter	See section 6.3.9.1
0x0001	Section Length	See section 6.3.9.2
0x0002	Flexible Filter Data	See section 6.3.9.3

6.3.9.1 Sub Module Type - MNG filter - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x107	Valid values are: 0x107 = MNG Filters

6.3.9.2 Section Length - 0x0001

The length of the section in words. Note that section length does not include a count for the section length word.

Bits	Field Name	Default NVM Value	Description
15:0	Section Length		Length in: 2 Bytes unit - 1 First Section -> Word: PFA MNG Filter -> Section Length Last Section -> Word: PFA MNG Filter -> Flexible Filter Data

6.3.9.3 Flexible Filter Data - 0x0002

Raw data module length: variable

Flexible Filter Data.

6.3.10 PFA PT configuration 0 Section Summary Table

Pass Through filters of port 0 PFA section header

Word Offset	Description	Reference
0x0000	Sub Module Type PT Module 0	See section 6.3.10.1
0x0001	Section Length	See section 6.3.10.2
0x0002 + 2*n, n=0...3	LAN IPv4 Address (LSB) MIPAF0	See section 6.3.10.3
0x0003 + 2*n, n=0...3	LAN IPv4 Address (MSB) MIPAF0	See section 6.3.10.4
0x000A + 2*n, n=0...15	LAN Flexible Filter Port	See section 6.3.10.5
0x000B + 2*n, n=0...15	LAN Flexible Filter Port - Modifiers	See section 6.3.10.6
0x002A + 1*n, n=0...7	LAN VLAN Filter	See section 6.3.10.7
0x0032	LAN MANC Value LSB	See section 6.3.10.8
0x0033	LAN MANC Value MSB	See section 6.3.10.9
0x0034	Receive Enable 1 (LRXEN1)	See section 6.3.10.10
0x0035	Receive Enable 2 (LRXEN2)	See section 6.3.10.11
0x0036	LAN MNGONLY LSB	See section 6.3.10.12
0x0037	LAN MNGONLY MSB	See section 6.3.10.13
0x0038 + 4*n, n=0...6	Manageability Decision Filters LSB	See section 6.3.10.14
0x0039 + 4*n, n=0...6	Manageability Decision Filters MSB	See section 6.3.10.15
0x003A + 4*n, n=0...6	Manageability Extended Decision Filters LSB	See section 6.3.10.16
0x003B + 4*n, n=0...6	Manageability Extended Decision Filters MSB	See section 6.3.10.17
0x0054 + 2*n, n=0...3	Manageability EtherType filter - METF - LSB	See section 6.3.10.18
0x0055 + 2*n, n=0...3	Manageability EtherType filter METF - MSB	See section 6.3.10.19
0x005C	ARP Response IPv4 Address (LSB)	See section 6.3.10.20
0x005D	ARP Response IPv4 Address (MSB)	See section 6.3.10.21
0x005E + 8*n, n=0...3	IPv6 Address Bytes 0-1	See section 6.3.10.22
0x005F + 8*n, n=0...3	IPv6 Address Bytes 2-3	See section 6.3.10.23
0x0060 + 8*n, n=0...3	IPv6 Address Bytes 4-5	See section 6.3.10.24
0x0061 + 8*n, n=0...3	IPv6 Address Bytes 6-7	See section 6.3.10.25
0x0062 + 8*n, n=0...3	IPv6 Address Bytes 8-9	See section 6.3.10.26
0x0063 + 8*n, n=0...3	IPv6 Address Bytes 10-11	See section 6.3.10.27
0x0064 + 8*n, n=0...3	IPv6 Address Bytes 12-13	See section 6.3.10.28
0x0065 + 8*n, n=0...3	IPv6 Address Bytes 14-15	See section 6.3.10.29
0x007E - 0x007F	Reserved	See section 6.3.10.30

6.3.10.1 Sub Module Type PT Module 0 - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x108	Valid values are: 0x108 = Pass-through Control Words Structure 0

6.3.10.2 Section Length - 0x0001

The length of the section in words. Note that section length does not include a count for the section length word.

Bits	Field Name	Default NVM Value	Description
15:0	Section Length		Length in: 2 Bytes unit - 1 First Section -> Word: PFA PT configuration 0 -> Section Length Last Section -> Word: PFA PT configuration 0 -> Reserved

6.3.10.3 LAN IPv4 Address (LSB) MIPAF0[n] (0x0002 + 2*n, n=0...3)

Bits	Field Name	Default NVM Value	Description
15:8	IPv4 Address, Byte 1	0xFF	
7:0	IPv4 Address, Byte 0	0xFF	

6.3.10.4 LAN IPv4 Address (MSB) MIPAF0[n] (0x0003 + 2*n, n=0...3)

Bits	Field Name	Default NVM Value	Description
15:8	IPv4 Address, Byte 3	0xFF	
7:0	IPv4 Address, Byte 2	0xFF	

6.3.10.5 LAN Flexible Filter Port[n] (0x000A + 2*n, n=0...15)

Bits	Field Name	Default NVM Value	Description
15:0	LAN UDP Flexible Filter Port0	0xFFFF	

6.3.10.6 LAN Flexible Filter Port - Modifiers[n] (0x000B + 2*n, n=0...15)

Bits	Field Name	Default NVM Value	Description
15:3	Reserved		
2	Source Destination	0x0	Valid values are: 0x0 = Destination 0x1 = Source
1	Accept TCP	0x0	If set, filter match if packet is TCP Valid values are: 0x0 = Filter 0x1 = Accept
0	Accept UDP	0x0	If set, filter match if packet is UDP Valid values are: 0x0 = Filter 0x1 = Accept

6.3.10.7 LAN VLAN Filter [n] (0x002A + 1*n, n=0...7)

Bits	Field Name	Default NVM Value	Description
15:12	Reserved	0x0	
11:0	VLAN Filter 0 Value	0xFFF	

6.3.10.8 LAN MANC Value LSB - 0x0032

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	

6.3.10.9 LAN MANC Value MSB - 0x0033

Bits	Field Name	Default NVM Value	Description
15:11	Reserved	0x0	
10	NET_TYPE	0x0	
9	FIXED_NET_TYPE	0x0	
8	Reserved	0x0	
7	EN_XSUM_FILTER	0x0	Should be used to set the RBCR.EN_XSUM_FILTERn bit (n is the port number).
6:0	Reserved	0x0	

6.3.10.10 Receive Enable 1 (LRXEN1) - 0x0034

Bits	Field Name	Default NVM Value	Description
15:9	Receive Enable byte 12	0x00	BMC SMBus target address
8	Reserved	0x0	
7	Enable BMC Dedicated	0x0	When set, the BMC will receive traffic sent to the MAC address defined in MMAH/MMAL[3]. The Firmware configures MDEF, MDEF_EXT (7) to dedicated MAC configured into MMAL/H (3) to implement this mode.
6	Reserved	0x1	
5:4	Notification method	0x0	This field is valid only in the section of port 0 and defines the notification method for all ports. Valid values are: 0x0 = SMBus alert 0x1 = Asynchronous notify 0x2 = Direct receive 0x3 = Reserved
3	Enable ARP Response	0x0	When set, the FW will offload ARP handling sent to the IP address defined in ARP Response IPv4 Address NVM words. The Firmware uses MDEF, MDEF_EXT (6) and MIPAF4[3] to implement this mode.
2	Enable Status Reporting	0x0	
1	Enable Receive All	0x0	
0	Enable Receive TCO	0x0	

6.3.10.11 Receive Enable 2 (LRXEN2) - 0x0035

Bits	Field Name	Default NVM Value	Description
15:8	Receive Enable byte 14	0x00	Alert Value
7:0	Receive Enable byte 13	0x00	Interface Data

6.3.10.12 LAN MNGONLY LSB - 0x0036

Bits	Field Name	Default NVM Value	Description
15:8	Reserved	0x0	
7:0	Exclusive to MNG	0x0	

6.3.10.13 LAN MNGONLY MSB - 0x0037

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	

6.3.10.14 Manageability Decision Filters LSB[n] (0x0038 + 4*n, n=0...6)

Bits	Field Name	Default NVM Value	Description
15:0	MDEF0_L	0x0	

6.3.10.15 Manageability Decision Filters MSB[n] (0x0039 + 4*n, n=0...6)

Bits	Field Name	Default NVM Value	Description
15:0	MDEF0_M	0x0	

6.3.10.16 Manageability Extended Decision Filters LSB[n] (0x003A + 4*n, n=0...6)

Bits	Field Name	Default NVM Value	Description
15:0	MDEFEXT0_L	0x0	

6.3.10.17 Manageability Extended Decision Filters MSB[n] (0x003B + 4*n, n=0...6)

Bits	Field Name	Default NVM Value	Description
15:0	MDEFEXT0_M	0x0	

6.3.10.18 Manageability EtherType filter - METF - LSB[n] (0x0054 + 2*n, n=0...3)

Bits	Field Name	Default NVM Value	Description
15:0	METF0_L	0x0000	Loaded to 16 LS bits of METF[0] register.

6.3.10.19 Manageability EtherType filter METF - MSB[n] (0x0055 + 2*n, n=0...3)

Bits	Field Name	Default NVM Value	Description
15:0	METF0_M	0x0000	Loaded to 16 MS bits of METF[0] register (reserved bits in the METF registers should be set in the NVM to the register default values).

6.3.10.20 ARP Response IPv4 Address (LSB) - 0x005C

Bits	Field Name	Default NVM Value	Description
15:8	Byte 1	0x00	FW use
7:0	Byte 0	0x00	FW use

6.3.10.21 ARP Response IPv4 Address (MSB) - 0x005D

Bits	Field Name	Default NVM Value	Description
15:8	Byte 3	0x00	FW use
7:0	Byte 2	0x00	FW use

6.3.10.22 IPv6 Address Bytes 0-1[n] (0x005E + 8*n, n=0...3)

Bits	Field Name	Default NVM Value	Description
15:0	IPv6 Address Bytes 0-1	0x0000	

6.3.10.23 IPv6 Address Bytes 2-3[n] (0x005F + 8*n, n=0...3)

Bits	Field Name	Default NVM Value	Description
15:0	IPv6 Address Bytes 2-3	0x0000	

6.3.10.24 IPv6 Address Bytes 4-5[n] (0x0060 + 8*n, n=0...3)

Bits	Field Name	Default NVM Value	Description
15:0	IPv6 Address Bytes 2-3	0x0000	

6.3.10.25 IPv6 Address Bytes 6-7[n] (0x0061 + 8*n, n=0...3)

Bits	Field Name	Default NVM Value	Description
15:0	IPv6 Address Bytes 6-7	0x0000	

6.3.10.26 IPv6 Address Bytes 8-9[n] (0x0062 + 8*n, n=0...3)

Bits	Field Name	Default NVM Value	Description
15:0	IPv6 Address Bytes 8-9	0x0000	

6.3.10.27 IPv6 Address Bytes 10-11[n] (0x0063 + 8*n, n=0...3)

Bits	Field Name	Default NVM Value	Description
15:0	IPv6 Address Bytes 10-11	0x0000	

6.3.10.28 IPv6 Address Bytes 12-13[n] (0x0064 + 8*n, n=0...3)

Bits	Field Name	Default NVM Value	Description
15:0	IPv6 Address Bytes 12-13	0x0000	

6.3.10.29 IPv6 Address Bytes 14-15[n] (0x0065 + 8*n, n=0...3)

Bits	Field Name	Default NVM Value	Description
15:0	IPv6 Address Bytes 14-15	0x0000	

6.3.10.30 Reserved (0x007E - 0x007F)

Bits	Field Name	Default NVM Value	Description
15:0	RESERVED	0x0000	

6.3.11 PFA PT configuration 1 Section Summary Table

Pass Through filters of port 1 PFA section header

Word Offset	Description	Reference
0x0000	Sub Module Type PT Module 1	See section 6.3.11.1
0x0001	Section Length	See section 6.3.11.2
0x0002 + 2*n, n=0...3	LAN IPv4 Address (LSB) MIPAF0	See section 6.3.11.3
0x0003 + 2*n, n=0...3	LAN IPv4 Address (MSB) MIPAF0	See section 6.3.11.4
0x000A + 2*n, n=0...15	LAN Flexible Filter Port	See section 6.3.11.5
0x000B + 2*n, n=0...15	LAN Flexible Filter Port - Modifiers	See section 6.3.11.6
0x002A + 1*n, n=0...7	LAN VLAN Filter	See section 6.3.11.7
0x0032	LAN MANC Value LSB	See section 6.3.11.8
0x0033	LAN MANC Value MSB	See section 6.3.11.9
0x0034	Receive Enable 1 (LRXEN1)	See section 6.3.11.10
0x0035	Receive Enable 2 (LRXEN2)	See section 6.3.11.11
0x0036	LAN MNGONLY LSB	See section 6.3.11.12
0x0037	LAN MNGONLY MSB	See section 6.3.11.13
0x0038 + 4*n, n=0...6	Manageability Decision Filters LSB	See section 6.3.11.14
0x0039 + 4*n, n=0...6	Manageability Decision Filters MSB	See section 6.3.11.15
0x003A + 4*n, n=0...6	Manageability Extended Decision Filters LSB	See section 6.3.11.16
0x003B + 4*n, n=0...6	Manageability Extended Decision Filters MSB	See section 6.3.11.17
0x0054 + 2*n, n=0...3	Manageability EtherType filter - METF - LSB	See section 6.3.11.18
0x0055 + 2*n, n=0...3	Manageability EtherType filter METF - MSB	See section 6.3.11.19
0x005C	ARP Response IPv4 Address (LSB)	See section 6.3.11.20
0x005D	ARP Response IPv4 Address (MSB)	See section 6.3.11.21
0x005E + 8*n, n=0...3	IPv6 Address Bytes 0-1	See section 6.3.11.22
0x005F + 8*n, n=0...3	IPv6 Address Bytes 2-3	See section 6.3.11.23
0x0060 + 8*n, n=0...3	IPv6 Address Bytes 4-5	See section 6.3.11.24
0x0061 + 8*n, n=0...3	IPv6 Address Bytes 6-7	See section 6.3.11.25
0x0062 + 8*n, n=0...3	IPv6 Address Bytes 8-9	See section 6.3.11.26
0x0063 + 8*n, n=0...3	IPv6 Address Bytes 10-11	See section 6.3.11.27
0x0064 + 8*n, n=0...3	IPv6 Address Bytes 12-13	See section 6.3.11.28
0x0065 + 8*n, n=0...3	IPv6 Address Bytes 14-15	See section 6.3.11.29
0x007E - 0x007F	Reserved	See section 6.3.11.30

6.3.11.1 Sub Module Type PT Module 1 - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x109	Valid values are: 0x109 = Pass-through Control Words Structure 1

6.3.11.2 Section Length - 0x0001

The length of the section in words. Note that section length does not include a count for the section length word.

For inner structure please See [section 6.3.10.2](#)

6.3.11.3 LAN IPv4 Address (LSB) MIPAF0[n] (0x0002 + 2*n, n=0...3)

For inner structure please See [section 6.3.10.3](#)

6.3.11.4 LAN IPv4 Address (MSB) MIPAF0[n] (0x0003 + 2*n, n=0...3)

For inner structure please See [section 6.3.10.4](#)

6.3.11.5 LAN Flexible Filter Port[n] (0x000A + 2*n, n=0...15)

For inner structure please See [section 6.3.10.5](#)

6.3.11.6 LAN Flexible Filter Port - Modifiers[n] (0x000B + 2*n, n=0...15)

For inner structure please See [section 6.3.10.6](#)

6.3.11.7 LAN VLAN Filter [n] (0x002A + 1*n, n=0...7)

For inner structure please See [section 6.3.10.7](#)

6.3.11.8 LAN MANC Value LSB - 0x0032

For inner structure please See [section 6.3.10.8](#)

6.3.11.9 LAN MANC Value MSB - 0x0033

For inner structure please See [section 6.3.10.9](#)

6.3.11.10 Receive Enable 1 (LRXEN1) - 0x0034

For inner structure please See [section 6.3.10.10](#)

6.3.11.11 Receive Enable 2 (LRXEN2) - 0x0035

For inner structure please See [section 6.3.10.11](#)

6.3.11.12 LAN MNGONLY LSB - 0x0036

For inner structure please See [section 6.3.10.12](#)

6.3.11.13 LAN MNGONLY MSB - 0x0037

For inner structure please See [section 6.3.10.13](#)

6.3.11.14 Manageability Decision Filters LSB[n] (0x0038 + 4*n, n=0...6)

For inner structure please See [section 6.3.10.14](#)

6.3.11.15 Manageability Decision Filters MSB[n] (0x0039 + 4*n, n=0...6)

For inner structure please See [section 6.3.10.15](#)

6.3.11.16 Manageability Extended Decision Filters LSB[n] (0x003A + 4*n, n=0...6)

For inner structure please See [section 6.3.10.16](#)

6.3.11.17 Manageability Extended Decision Filters MSB[n] (0x003B + 4*n, n=0...6)

For inner structure please See [section 6.3.10.17](#)

6.3.11.18 Manageability EtherType filter - METF - LSB[n] (0x0054 + 2*n, n=0...3)

For inner structure please See [section 6.3.10.18](#)

6.3.11.19 Manageability EtherType filter METF - MSB[n] (0x0055 + 2*n, n=0...3)

For inner structure please See [section 6.3.10.19](#)

6.3.11.20 ARP Response IPv4 Address (LSB) - 0x005C

For inner structure please See [section 6.3.10.20](#)

6.3.11.21 ARP Response IPv4 Address (MSB) - 0x005D

For inner structure please See [section 6.3.10.21](#)

6.3.11.22 IPv6 Address Bytes 0-1[n] (0x005E + 8*n, n=0...3)

For inner structure please See [section 6.3.10.22](#)

6.3.11.23 IPv6 Address Bytes 2-3[n] (0x005F + 8*n, n=0...3)

For inner structure please See [section 6.3.10.23](#)

6.3.11.24 IPv6 Address Bytes 4-5[n] (0x0060 + 8*n, n=0...3)

For inner structure please See [section 6.3.10.24](#)

6.3.11.25 IPv6 Address Bytes 6-7[n] (0x0061 + 8*n, n=0...3)

For inner structure please See [section 6.3.10.25](#)

6.3.11.26 IPv6 Address Bytes 8-9[n] (0x0062 + 8*n, n=0...3)

For inner structure please See [section 6.3.10.26](#)

6.3.11.27 IPv6 Address Bytes 10-11[n] (0x0063 + 8*n, n=0...3)

For inner structure please See [section 6.3.10.27](#)

6.3.11.28 IPv6 Address Bytes 12-13[n] (0x0064 + 8*n, n=0...3)

For inner structure please See [section 6.3.10.28](#)

6.3.11.29 IPv6 Address Bytes 14-15[n] (0x0065 + 8*n, n=0...3)

For inner structure please See [section 6.3.10.29](#)

6.3.11.30 Reserved (0x007E - 0x007F)

For inner structure please See [section 6.3.10.30](#)

6.3.12 Original EETrack ID Section Summary Table

Original EETrack ID PFA section

Word Offset	Description	Reference
0x0000	Sub Module Type Original EETrackID	See section 6.3.12.1
0x0001	Length	See section 6.3.12.2
0x0002	Original EETrackID - MSB	See section 6.3.12.3
0x0003	Original EETrackID - LSB	See section 6.3.12.4

6.3.12.1 Sub Module Type Original EETrackID - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x10C	Valid values are: 0x10C = Original EETrack ID

6.3.12.2 Length - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit - 2 First Section -> Word: Original EETrack ID -> Sub Module Type Original EETrackID Last Section -> Word: Original EETrack ID -> Original EETrackID - LSB

6.3.12.3 Original EETrackID - MSB - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	EETrackID - MSB	0x0	

6.3.12.4 Original EETrackID - LSB - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	EETrackID - LSB	0x0	

6.3.13 IBA capabilities Module Section Summary Table

IBA PFA section

Word Offset	Description	Reference
0x0000	Sub Module Type IBA Capabilities	See section 6.3.13.1
0x0001	Length	See section 6.3.13.2
0x0002	IBA Capabilities	See section 6.3.13.3

6.3.13.1 Sub Module Type IBA Capabilities - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x33	Valid values are: 0x33 = IBA Capabilities

6.3.13.2 Length - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit - 2 First Section -> Word: IBA capabilities Module -> Sub Module Type IBA Capabilities Last Section -> Word: IBA capabilities Module -> IBA Capabilities

6.3.13.3 IBA Capabilities - 0x0002

This word of the NVM is used to enumerate the boot technologies that have been programmed into the Flash. This is updated by Flash configuration tools and is not updated or read by IBA.

Bits	Field Name	Default NVM Value	Description
15:14	Signature	0x1	Signature. Must be set to 01b to indicate that this word has been programmed by the agent or other configuration software.
13	Allow PXE Disable	0x0	If set to 0b, PXE is always load regardless the settings in bits 2:0 in Main Setup Options word. If set to 1b, PXE is loaded when bits 2:0 in Main Setup Options are set to 0 (PXE enabled on this port). Valid values are: 0x0 = Ignore PXE Disable 0x1 = Allow PXE Disable
12:5	Reserved	0x0	Reserved. Must be 0.
4	iSCSI boot	0x0	iSCSI is present if set to 1. Valid values are: 0x0 = Not Present 0x1 = Present
3	efi ebd driver	0x0	EFI UNDI driver is present if set to 1. Valid values are: 0x0 = Not Present 0x1 = Present

Bits	Field Name	Default NVM Value	Description
2	RPL	0x0	RPL module is present if set to 1. Reserved bit for devices Valid values are: 0x0 = Not Present 0x1 = Present
1	PXE/UNDI Driver	0x1	PXE UNDI driver is present if set to 1. Valid values are: 0x0 = Not Present 0x1 = Present
0	PXE Base Code	0x1	PXE Base Code is present if set to 1. Valid values are: 0x0 = Not Present 0x1 = Present

6.3.14 PXE Setup Options Module Section Summary Table

IBA PFA section

Word Offset	Description	Reference
0x0000	Sub Module Type PXE setup	See section 6.3.14.1
0x0001	Length	See section 6.3.14.2
0x0002 + 1*n, n=0...1	Setup Options PCI Function	See section 6.3.14.3

6.3.14.1 Sub Module Type PXE setup - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x30	Valid values are: 0x30 = PXE Setup Options

6.3.14.2 Length - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit - 2 First Section -> Word: PXE Setup Options Module -> Sub Module Type PXE setup Last Section -> Word: PXE Setup Options Module -> Setup Options PCI Function

6.3.14.3 Setup Options PCI Function[n] (0x0002 + 1*n, n=0...1)

The main setup options for PF n are stored in this word. These options are those that can be changed by the user using the Control-S setup menu.

Bits	Field Name	Default NVM Value	Description
15:13	Reserved	0x00	Reserved. Must be 0x0.
12:10	FSD	0x0	Bits 12-10 control forcing speed and duplex during driver operation. Valid values are: 000b = Auto-negotiate. 001b = 10 Mb/s half duplex. 010b = 100 Mb/s half duplex. 011b = Not valid (treated as 000b). 100b = 10 Mb/s full duplex. 101b = 100 Mb/s full duplex. 111b = 1000 Mb/s full duplex. Only applicable for copper-based adapters. Not applicable to 10 GbE. Default value is 000b. Valid values are: 0x0 = Auto-negotiate 0x1 = 10Mbps Half Duplex 0x2 = 100Mbps Half Duplex 0x4 = 10Mbps Full Duplex 0x5 = 100Mbps Full Duplex 0x7 = 1000Mbps Full Duplex
9	Reserved	0x0	Reserved to legacy OS Wakeup Support.

Bits	Field Name	Default NVM Value	Description
8	DSM	0x1	Display Setup Message. If the bit is set to 1b, the Press Control-S message is displayed after the title message. Default value is 1b.
7:6	PT	0x0	Prompt Time. These bits control how long the Press Control-S setup prompt message is displayed, if enabled by DIM. 00b = 2 seconds (default). 01b = 3 seconds. 10b = 5 seconds. 11b = 0 seconds. Note: The Ctrl-S message is not displayed if 0 seconds prompt time is selected. Valid values are: 0x0 = 2 sec 0x1 = 3 sec 0x2 = 5 sec 0x3 = 0 sec
5	iSCSI Boot disabled	0x0	When this bit is set and adapter port is neither iSCSI primary nor secondary, Setup code shall not be loaded. Otherwise iSCSI banner and Setup menu should be accessible as in current design. This bit shall be changed at factory level and not be altered by any end-customer tools. Note: For regular (non Dell combo) NICs and LOM design this bit should be always cleared in NVM image otherwise iSCSI setup will not be accessible for the user. Valid values are: 0x0 = Enabled 0x1 = Disabled
4:3	DBS	0x0	Default Boot Selection. These bits select which device is the default boot device. These bits are only used if the agent detects that the BIOS does not support boot order selection or if the MODE field of word 31h is set to MODE_LEGACY. 00 = Network boot, then local boot (default) 01 = Local boot, then network boot 10 = Network boot only 11 = Local boot only Valid values are: 0x0 = Network then local 0x1 = Local then network 0x2 = Network only 0x3 = Local only
2:0	PS	0x0	Protocol Select. These bits select the active boot protocol. 00b = PXE (default value). 01b = RPL (only if RPL is in the Flash). 10b = iSCSI Boot primary port (only if iSCSI Boot is using this adapter). 11b = iSCSI Boot secondary port (only if iSCSI Boot is using this adapter). Only the default value of 00b should be initially programmed into the adapter; other values should only be set by configuration utilities. Valid values are: 0x0 = PXE (default value). 0x1 = Boot Disabled 0x2 = iSCSI Primary 0x3 = iSCSI Secondary 0x4 = Reserved 0x5 = Reserved 0x6 = Reserved 0x7 = Reserved

6.3.15 PXE Configuration Customization Options Module Section Summary Table

IBA PFA section

Word Offset	Description	Reference
0x0000	Sub Module Type PXE Configuration Customization Options	See section 6.3.15.1
0x0001	Length	See section 6.3.15.2
0x0002 + 1*n, n=0...1	Configuration Customization Options PCI Function	See section 6.3.15.3

6.3.15.1 Sub Module Type PXE Configuration Customization Options - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x31	Valid values are: 0x31 = PXE Configuration Customization Options

6.3.15.2 Length - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit - 2 First Section -> Word: PXE Configuration Customization Options Module -> Sub Module Type PXE Configuration Customization Options Last Section -> Word: PXE Configuration Customization Options Module -> Configuration Customization Options PCI Function

6.3.15.3 Configuration Customization Options PCI Function[n] (0x0002 + 1*n, n=0...1)

It contains settings that can be programmed by an OEM or network administrator to customize the operation of the software. These settings cannot be changed from within the Control- S setup menu. The lower byte contains settings that would typically be configured by a network administrator using an external utility; these settings generally control which setup menu options are changeable. The upper byte is generally settings that would be used by an OEM to control the operation of the agent in a LOM environment, although there is nothing in the agent to prevent their use on a NIC implementation. The default value for this word is 0x4000.

Bits	Field Name	Default NVM Value	Description
15:14	Signature	0x1	Signature. Must be set to 01 to indicate that this word has been programmed by the agent or other configuration software.
13:12	Reserved	0x0	Reserved. Must be 0.

Bits	Field Name	Default NVM Value	Description
11	Continuous Retry	0x0	Selects Continuous Retry operation. If this bit is set, IBA does NOT transfer control back to the BIOS if it fails to boot due to a network error (such as failure to receive DHCP replies). Instead, it restarts the PXE boot process again. If this bit is set, the only way to cancel PXE boot is for the user to press ESC on the keyboard. Retry is not attempted due to hardware conditions such as an invalid NVM checksum or failing to establish link. Default value is 0b. Valid values are: 0x0 = Disable 0x1 = Enable
10:8	Operating mode	0x0	Selects the agent's boot order setup mode. This field changes the agent default behavior in order to make it compatible with systems that do not completely support the BBS and PnP Expansion ROM standards. Valid values and their meanings are: 000b = Normal behavior. The agent attempts to detect BBS and PnP Expansion ROM support as it normally does. 001b = Force legacy mode. The agent does not attempt to detect BBS or PnP Expansion ROM supports in the BIOS and assumes the BIOS is not compliant. The user can change the BIOS boot order in the Setup Menu. 010b = Force BBS mode. The agent assumes the BIOS is BBS compliant, even though it might not be detected as such by the agent's detection code. The user CANNOT change the BIOS boot order in the Setup Menu. 011b = Force PnP Int18 mode. The agent assumes the BIOS allows boot order setup for PnP Expansion ROMs and hooks interrupt 0x18 (to inform the BIOS that the agent is a bootable device) in addition to registering as a BBS IPL device. The user CANNOT change the BIOS boot order in the Setup Menu. 100b = Force PnP Int19 mode. The agent assumes the BIOS allows boot order setup for PnP Expansion ROMs and hook interrupt 0x19 (to inform the BIOS that the agent is a bootable device) in addition to registering as a BBS IPL device. The user CANNOT change the BIOS boot order in the Setup Menu. 101b = Reserved for future use. If specified, is treated as a value of 000b. 110b = Reserved for future use. If specified, is treated as a value of 000b. 111b = Reserved for future use. If specified, is treated as a value of 000b. Valid values are: 0x0 = Normal Mode 0x1 = Force Legacy 0x2 = Force BBS 0x3 = Force PnP Int18 0x4 = Force PnP Int19
7:6	Reserved	0x0	Reserved. Must be 0.
5	Disable Flash Update	0x0	Disable Flash Update. If this bit is set to 1, the user is not allowed to update the flash image using PROSet. Default value is 0. Valid values are: 0x0 = Enable Flash Update 0x1 = Disable Flash Update
4	Disable Legacy OS Wakeup Menu	0x0	Disable Legacy Wakeup Support. If this bit is set to 1, the user is not allowed to change the Legacy OS Wakeup Support menu option. Default value is 0. Valid values are: 0x0 = Enable - Legacy Wakeup Support 0x1 = Disable - Legacy Wakeup Support
3	Disable Boot Selection Menu	0x0	Disable Boot Selection. If this bit is set to 1, the user is not allowed to change the boot order menu option. Default value is 0. Valid values are: 0x0 = Enable 0x1 = Disable
2	Disable Protocol Selection Menu	0x0	Disable Protocol Select. If set to 1, the user is not allowed to change the boot protocol. Default value is 0. Valid values are: 0x0 = Enable 0x1 = Disable

Bits	Field Name	Default NVM Value	Description
1	Disable Title Message Display	0x0	Disable Title Message. If this bit is set to 1, the title message displaying the version of the Boot Agent is suppressed; the Control-S message is also suppressed. This is for OEMs who do not wish the boot agent to display any messages at system boot. Default value is 0. Valid values are: 0x0 = Enable 0x1 = Disable
0	Setup Menu	0x0	Disable Setup Menu. If this bit is set to 1, the user is not allowed to invoke the setup menu by pressing Control-S. In this case, the NVM may only be changed via an external program. Default value is 0. Valid values are: 0x0 = Enable 0x1 = Disable

6.3.16 PXE version Module Section Summary Table

IBA PFA section

Word Offset	Description	Reference
0x0000	Sub Module Type PXE version	See section 6.3.16.1
0x0001	Length	See section 6.3.16.2
0x0002	PXE Version	See section 6.3.16.3

6.3.16.1 Sub Module Type PXE version - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x32	Valid values are: 0x32 = PXE version

6.3.16.2 Length - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit - 2 First Section -> Word: PXE version Module -> Sub Module Type PXE version Last Section -> Word: PXE version Module -> PXE Version

6.3.16.3 PXE Version - 0x0002

Word 0x32 of the NVM is used to store the version of the boot agent that is stored in the Flash image. When the Boot Agent loads, it can check this value to determine if any first-time configuration needs to be performed. The agent then updates this word with its version. Some diagnostic tools to report the version of the Boot Agent in the Flash also read this word.

Bits	Field Name	Default NVM Value	Description
15:12	Major Version	0x0	PXE Boot Agent Major Version. Default value is 0.
11:8	Minor Version	0x0	PXE Boot Agent Minor Version. Default value is 0.
7:0	Build Number	0x0	PXE Boot Agent Build Number. Default value is 0.

6.3.17 VLAN Module Section Summary Table

IBA PFA section

Word Offset	Description	Reference
0x0000	Sub Module Type VLAN	See section 6.3.17.1
0x0001	Length	See section 6.3.17.2
0x0002	VLAN block signature	See section 6.3.17.3
0x0003	Structure Version and Size	See section 6.3.17.4
0x0004	Port 0 VLAN tag	See section 6.3.17.5
0x0005	Port 1 VLAN tag	See section 6.3.17.6

6.3.17.1 Sub Module Type VLAN - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x37	Valid values are: 0x37 = VLAN Configuration

6.3.17.2 Length - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit - 2 First Section -> Word: VLAN Module -> Sub Module Type VLAN Last Section -> Word: VLAN Module -> Port 1 VLAN tag

6.3.17.3 VLAN block signature - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	VLAN block signature	0x4C56	"V"=0x56, "L" = 0x4C

6.3.17.4 Structure Version and Size - 0x0003

Bits	Field Name	Default NVM Value	Description
15:8	Structure Version and Size	0x0C	Total byte size of the configuration block. This is 0x6 for 1-port adapter, 0x8 for 2-port adapter and 0xC for 4-port adapter (the default value).
7:0	Structure version	0x01	The version of this structure. Should be set to 0x1.

6.3.17.5 Port 0 VLAN tag - 0x0004

Bits	Field Name	Default NVM Value	Description
15:13	VLAN Priority	0x0	The value of VLAN priority (0-7).
12	Reserved	0x0	This field is reserved and must be set to 0x0.
11:0	VLAN tag ID	0x0	The value of VLAN ID (1-4095). 0 means no VLAN configured for port.

6.3.17.6 Port 1 VLAN tag - 0x0005

For inner structure please See [section 6.3.17.5](#)

6.3.18 Boot Configuration Block Section Summary Table

Preserved Fields Area Header

Word Offset	Description	Reference
0x0000	Sub Module Type	See section 6.3.18.1
0x0001	Length	See section 6.3.18.2
0x0002	Combo Image Version High	See section 6.3.18.3
0x0003	Combo Image Version Low	See section 6.3.18.4

6.3.18.1 Sub Module Type - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x132	Valid values are: 0x132 = CIVD

6.3.18.2 Length - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit - 2 First Section -> Word: Boot Configuration Block -> Sub Module Type Last Section -> Word: Boot Configuration Block -> Combo Image Version Low

6.3.18.3 Combo Image Version High - 0x0002

Bits	Field Name	Default NVM Value	Description
15:8	Major	0x0	
7:0	Build	0x0	

6.3.18.4 Combo Image Version Low - 0x0003

Bits	Field Name	Default NVM Value	Description
15:8	Build	0x0	
7:0	Patch	0x0	

6.3.19 PBA header Section Summary Table

Preserved Fields Area Header

Word Offset	Description	Reference
0x0000	Sub Module Type - PBA	See section 6.3.19.1
0x0001	Length	See section 6.3.19.2

6.3.19.1 Sub Module Type - PBA - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x16	Valid values are: 0x16 = PBA block

6.3.19.2 Length - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit First Section -> Word: PBA Block -> PBA Section Length Last Section -> Word: PBA Block -> Word5

6.3.20 PBA Block Section Summary Table

The PBA block contains the complete PBA number including the dash and the first digit of the 3-digit suffix.

Word Offset	Description	Reference
0x0000	PBA Section Length	See section 6.3.20.1
0x0001	Word1	See section 6.3.20.2
0x0002	Word2	See section 6.3.20.3
0x0003	Word3	See section 6.3.20.4
0x0004	Word4	See section 6.3.20.5
0x0005	Word5	See section 6.3.20.6

6.3.20.1 PBA Section Length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	PBA Section Length field	0x6	Length in words of the PBA Block.

6.3.20.2 Word1 - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Word1 field		PBA Number stored in hexadecimal ASCII values.

6.3.20.3 Word2 - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	Word2 field		PBA Number stored in hexadecimal ASCII values.

6.3.20.4 Word3 - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	Word3 field		PBA Number stored in hexadecimal ASCII values.

6.3.20.5 Word4 - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	Word4 field		PBA Number stored in hexadecimal ASCII values.

6.3.20.6 Word5 - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	Word5 field		PBA Number stored in hexadecimal ASCII values.

6.3.21 PCIR Registers PFA Autoload Module Section Summary Table

Preserved Fields Area Header

Word Offset	Description	Reference
0x0000	PCIR Registers Autoload Type	See section 6.3.21.1
0x0001	Module Length	See section 6.3.21.2
0x0002	Address Low at PCI_CNF2	See section 6.3.21.3
0x0003	Address High at PCI_CNF2	See section 6.3.21.4
0x0004	PCI_CNF2 low	See section 6.3.21.5
0x0005	PCI_CNF2 high	See section 6.3.21.6
0x0006	Address Low at PCI_LBARCTRL	See section 6.3.21.7
0x0007	Address High at PCI_LBARCTRL	See section 6.3.21.8
0x0008	PCI_LBARCTRL L	See section 6.3.21.9
0x0009	PCI_LBARCTRL H	See section 6.3.21.10
0x000A	Address Low at PCI_SERL	See section 6.3.21.11
0x000B	Address High at PCI_SERL	See section 6.3.21.12
0x000C	PCI_SERL 1	See section 6.3.21.13
0x000D	PCI_SERL 2	See section 6.3.21.14
0x000E	Address Low at PCI_SERH	See section 6.3.21.15
0x000F	Address High at PCI_SERH	See section 6.3.21.16
0x0010	PCI_SERL 3	See section 6.3.21.17
0x0011	PCI_SERL 4	See section 6.3.21.18
0x0012	Address Low at PCI_CAPCTRL	See section 6.3.21.19
0x0013	Address High at PCI_CAPCTRL	See section 6.3.21.20
0x0014	PCI_CAPCTRL L	See section 6.3.21.21
0x0015	PCI_CAPCTRL H	See section 6.3.21.22
0x0016	Address Low at PCI_CAPSUP	See section 6.3.21.23
0x0017	Address High at PCI_CAPSUP	See section 6.3.21.24
0x0018	PCI_CAPSUP L	See section 6.3.21.25
0x0019	PCI_CAPSUP H	See section 6.3.21.26
0x001A	Address Low at PCI_DBGCTL	See section 6.3.21.27
0x001B	Address High at PCI_DBGCTL	See section 6.3.21.28
0x001C	PCI_DBGCTL L	See section 6.3.21.29
0x001D	PCI_DBGCTL H	See section 6.3.21.30
0x001E	Address Low at PCI_UPADD	See section 6.3.21.31
0x001F	Address High at PCI_UPADD	See section 6.3.21.32
0x0020	PCI_UPADD L	See section 6.3.21.33
0x0021	PCI_UPADD H	See section 6.3.21.34

Word Offset	Description	Reference
0x0022	Address Low at PCI_SUBSYSID	See section 6.3.21.35
0x0023	Address High at PCI_SUBSYSID	See section 6.3.21.36
0x0024	PCI_SUBSYSID L	See section 6.3.21.37
0x0025	PCI_SUBSYSID H	See section 6.3.21.38
0x0026	Address Low at PCI_VFSUP	See section 6.3.21.39
0x0027	Address High at PCI_VFSUP	See section 6.3.21.40
0x0028	PCI_VFSUP L	See section 6.3.21.41
0x0029	PCI_VFSUP H	See section 6.3.21.42
0x002A	Address Low at TPH_CTRL	See section 6.3.21.43
0x002B	Address High at TPH_CTRL	See section 6.3.21.44
0x002C	TPH_CTRL L	See section 6.3.21.45
0x002D	TPH_CTRL H	See section 6.3.21.46
0x002E	Address Low at PCI_LINKCAP	See section 6.3.21.47
0x002F	Address High at PCI_LINKCAP	See section 6.3.21.48
0x0030	PCI_LINKCAP L	See section 6.3.21.49
0x0031	PCI_LINKCAP H	See section 6.3.21.50
0x0032	Address Low at PCI_GLBL_CNF	See section 6.3.21.51
0x0033	Address High at PCI_GLBL_CNF	See section 6.3.21.52
0x0034	PCI_GLBL_CNF L	See section 6.3.21.53
0x0035	PCI_GLBL_CNF H	See section 6.3.21.54
0x0036	Address Low at PCI_VENDORID	See section 6.3.21.55
0x0037	Address High at PCI_VENDORID	See section 6.3.21.56
0x0038	PCI_VENDORID L	See section 6.3.21.57
0x0039	PCI_VENDORID H	See section 6.3.21.58
0x003A	Address Low at PCI_PFDEVID	See section 6.3.21.59
0x003B	Address High at PCI_PFDEVID	See section 6.3.21.60
0x003C	PCI_PFDEVID Low (LAN)	See section 6.3.21.61
0x003D	PCI_PFDEVID High (SAN)	See section 6.3.21.62
0x003E	Address Low at PCI_CNF	See section 6.3.21.63
0x003F	Address High at PCI_CNF	See section 6.3.21.64
0x0040	PCI_CNF Low	See section 6.3.21.65
0x0041	PCI_CNF High	See section 6.3.21.66
0x0042	Address Low at PCI_VFDEVID	See section 6.3.21.67
0x0043	Address High at PCI_VFDEVID	See section 6.3.21.68
0x0044	PCI_VFDEVID Low (LAN)	See section 6.3.21.69
0x0045	PCI_VFDEVID High (SAN)	See section 6.3.21.70
0x0046	Address Low at PCI_CLASS	See section 6.3.21.71
0x0047	Address High at PCI_CLASS	See section 6.3.21.72
0x0048	PCI_CLASS Low	See section 6.3.21.73

Word Offset	Description	Reference
0x0049	PCI_CLASS High	See section 6.3.21.74
0x004A	Address Low at PCI_FUNC2	See section 6.3.21.75
0x004B	Address High at PCI_FUNC2	See section 6.3.21.76
0x004C	PCI_FUNC2 Low	See section 6.3.21.77
0x004D	PCI_FUNC2 High	See section 6.3.21.78
0x004E	Address Low at PCI_PFDEVID	See section 6.3.21.79
0x004F	Address High at PCI_PFDEVID	See section 6.3.21.80
0x0050	PCI_PFDEVID Low (LAN)	See section 6.3.21.81
0x0051	PCI_PFDEVID High (SAN)	See section 6.3.21.82
0x0052	Address Low at PCI_CNF	See section 6.3.21.83
0x0053	Address High at PCI_CNF	See section 6.3.21.84
0x0054	PCI_CNF Low	See section 6.3.21.85
0x0055	PCI_CNF High	See section 6.3.21.86
0x0056	Address Low at PCI_VFDEVID	See section 6.3.21.87
0x0057	Address High at PCI_VFDEVID	See section 6.3.21.88
0x0058	PCI_VFDEVID Low (LAN)	See section 6.3.21.89
0x0059	PCI_VFDEVID High (SAN)	See section 6.3.21.90
0x005A	Address Low at PCI_CLASS	See section 6.3.21.91
0x005B	Address High at PCI_CLASS	See section 6.3.21.92
0x005C	PCI_CLASS Low	See section 6.3.21.93
0x005D	PCI_CLASS High	See section 6.3.21.94
0x005E	Address Low at PCI_FUNC2	See section 6.3.21.95
0x005F	Address High at PCI_FUNC2	See section 6.3.21.96
0x0060	PCI_FUNC2 Low	See section 6.3.21.97
0x0061	PCI_FUNC2 High	See section 6.3.21.98

6.3.21.1 PCIR Registers Autoload Type - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x113	Valid values are: 0x113 = PCIR Registers Autoload

6.3.21.2 Module Length - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 2 First Section -> Word: PCIR Registers PFA Autoload Module -> PCIR Registers Autoload Type Last Section -> Word: PCIR Registers PFA Autoload Module -> PCI_FUNC2 High

6.3.21.3 Address Low at PCI_CNF2 - 0x0002

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_CNF2	0x110F8	
3:0	Type	0x1	

6.3.21.4 Address High at PCI_CNF2 - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_CNF2		

6.3.21.5 PCI_CNF2 low - 0x0004

MSI-x and cache line configuration

Bits	Field Name	Default NVM Value	Description
15:0	PCI_CNF2 low	0x400040FC	

6.3.21.6 PCI_CNF2 high - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	PCI_CNF2 high		

6.3.21.7 Address Low at PCI_LBARCTRL - 0x0006

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_LBARCTRL	0x11048	
3:0	Type	0x1	

6.3.21.8 Address High at PCI_LBARCTRL - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_LBARCTRL		

6.3.21.9 PCI_LBARCTRL L - 0x0008

PF BAR registers configuration

Bits	Field Name	Default NVM Value	Description
15:14	RESERVED	0x0	
13:11	EXROM_BAR_SIZE	0x3	This field indicates the size of the expansion ROM BAR as = 64KB x (2 ** EXROM_BAR_SIZE). 000b corresponds to a 64KB size. 111b corresponds to a 8MB size. Default value is 512KB.
10:9	RESERVED	0x0	
8:6	FL_BAR_SIZE	0x7	This field indicates the size of the external Flash as = 64KB x (2 ** FL_BAR_SIZE) as used to define exposure in the BAR. The following values are supported: 101b - 2MB 110b - 4MB 111b - 8MB Else - Reserved
5:4	RESERVED	0x0	
3	Reserved	0x1	
2	CSRSIZE	0x1	The CSRSIZE and FL_BAR_SIZE fields define the usable FLASH size and CSR mapping window size.
1	BAR32	0x0	BAR 32bit Enable. 0b - 64bit BAR addressing mode is selected. 1b - 32bit BARs are enabled.
0	PREFBAR	0x1	Prefetchable bit indication in the memory BARs (should be set when 64bit BARs are used) 0b - BARs are marked as non prefetchable 1b - BARs are marked as prefetchable

6.3.21.10 PCI_LBARCTRL H - 0x0009

Bits	Field Name	Default NVM Value	Description
15:0	PCI_LBARCTRL H	0x0	

6.3.21.11 Address Low at PCI_SERL - 0x000A

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_SERL	0x110F0	
3:0	Type	0x1	

6.3.21.12 Address High at PCI_SERL - 0x000B

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_SERL		

6.3.21.13 PCI_SERL 1 - 0x000C

Serial MAC Address Byte0 [LSB] and Byte1

Bits	Field Name	Default NVM Value	Description
15:0	PCI_SERL 1	0x000	Serial MAC Address Byte0 [LSB] and Byte1 Note: This field is preserved by Intel NVM update tool.

6.3.21.14 PCI_SERL 2 - 0x000D

Serial MAC Address Byte2 [LSB] and Byte3

Bits	Field Name	Default NVM Value	Description
15:0	PCI_SERL 2	0xc9	Serial MAC Address Byte2 [LSB] and Byte3 Note: This field is preserved by Intel NVM update tool.

6.3.21.15 Address Low at PCI_SERH - 0x000E

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_SERH	0x11078	
3:0	Type	0x1	

6.3.21.16 Address High at PCI_SERH - 0x000F

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_SERH		

6.3.21.17 PCI_SERL 3 - 0x0010

Serial MAC Address Byte4 [LSB] and Byte5

Bits	Field Name	Default NVM Value	Description
15:0	PCI_SERL 3	0x0	Serial MAC Address Byte4 [LSB] and Byte5 Note: This field is preserved by Intel NVM update tool.

6.3.21.18 PCI_SERL 4 - 0x0011

Bits	Field Name	Default NVM Value	Description
15:0	PCI_SERL 4	0x0	Note: This field is preserved by Intel NVM update tool.

6.3.21.19 Address Low at PCI_CAPCTRL - 0x0012

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_CAPCTRL	0x11080	
3:0	Type	0x1	

6.3.21.20 Address High at PCI_CAPCTRL - 0x0013

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_CAPCTRL		

6.3.21.21 PCI_CAPCTRL L - 0x0014

VPD_EN

Bits	Field Name	Default NVM Value	Description
15:1	PCI_CAPCTRL L	0x0	
0	VPD_EN	0x1	

6.3.21.22 PCI_CAPCTRL H - 0x0015

Bits	Field Name	Default NVM Value	Description
15:0	PCI_CAPCTRL H	0x0	

6.3.21.23 Address Low at PCI_CAPSUP - 0x0016

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_CAPSUP	0x11088	
3:0	Type	0x1	

6.3.21.24 Address High at PCI_CAPSUP - 0x0017

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_CAPSUP		

6.3.21.25 PCI_CAPSUP L - 0x0018

PCIe Capabilities exposed

Bits	Field Name	Default NVM Value	Description
15:0	PCI_CAPSUP L	0xC01C4EFF	

6.3.21.26 PCI_CAPSUP H - 0x0019

Bits	Field Name	Default NVM Value	Description
15:0	PCI_CAPSUP H		

6.3.21.27 Address Low at PCI_DBGCTL - 0x001A

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_DBGCTL	0x110E4	
3:0	Type	0x1	

6.3.21.28 Address High at PCI_DBGCTL - 0x001B

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_DBGCTL		

6.3.21.29 PCI_DBGCTL L - 0x001C

Allow access to VF config space

Bits	Field Name	Default NVM Value	Description
15:0	PCI_DBGCTL L	0x0	

6.3.21.30 PCI_DBGCTL H - 0x001D

Bits	Field Name	Default NVM Value	Description
15:0	PCI_DBGCTL H		

6.3.21.31 Address Low at PCI_UPADD - 0x001E

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_UPADD	0x110E8	
3:0	Type	0x1	

6.3.21.32 Address High at PCI_UPADD - 0x001F

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_UPADD		

6.3.21.33 PCI_UPADD L - 0x0020

Upper Address accessible

Bits	Field Name	Default NVM Value	Description
15:1	ADDRESS	0x0	
0	MODE	0x0	

6.3.21.34 PCI_UPADD H - 0x0021

Bits	Field Name	Default NVM Value	Description
15:0	ADDRESS		

6.3.21.35 Address Low at PCI_SUBSYSID - 0x0022

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PFPCI_SUBSYSID	0x11058	
3:0	Type	0x1	

6.3.21.36 Address High at PCI_SUBSYSID - 0x0023

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PFPCI_SUBSYSID		

6.3.21.37 PCI_SUBSYSID L - 0x0024

Sub-Vendor ID

Bits	Field Name	Default NVM Value	Description
15:0	SUB_VEN_ID	0x8086	Note: This field is preserved by Intel NVM update tool.

6.3.21.38 PCI_SUBSYSID H - 0x0025

Sub-System ID

Bits	Field Name	Default NVM Value	Description
15:0	SUB_ID	0x0	Note: This field is preserved by Intel NVM update tool.

6.3.21.39 Address Low at PCI_VFSUP - 0x0026

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_VFSUP	0x110A8	
3:0	Type	0x1	

6.3.21.40 Address High at PCI_VFSUP - 0x0027

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_VFSUP		

6.3.21.41 PCI_VFSUP L - 0x0028

PF BAR registers configuration

Bits	Field Name	Default NVM Value	Description
15:0	PCI_VFSUP L	0x0	

6.3.21.42 PCI_VFSUP H - 0x0029

Bits	Field Name	Default NVM Value	Description
15:0	PCI_VFSUP H		

6.3.21.43 Address Low at TPH_CTRL - 0x002A

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of TPH_CTRL	0x11074	
3:0	Type	0x1	

6.3.21.44 Address High at TPH_CTRL - 0x002B

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of TPH_CTRL		

6.3.21.45 TPH_CTRL L - 0x002C

TPH Control Register

Bits	Field Name	Default NVM Value	Description
15:0	TPH_CTRL L	0x00001800	

6.3.21.46 TPH_CTRL H - 0x002D

Bits	Field Name	Default NVM Value	Description
15:0	TPH_CTRL H		

6.3.21.47 Address Low at PCI_LINKCAP - 0x002E

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_LINKCAP	0x11090	
3:0	Type	0x1	

6.3.21.48 Address High at PCI_LINKCAP - 0x002F

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_LINKCAP		

6.3.21.49 PCI_LINKCAP L - 0x0030

PCIe Link Capabilities

Bits	Field Name	Default NVM Value	Description
15:0	PCI_LINKCAP L	0x00000E80	

6.3.21.50 PCI_LINKCAP H - 0x0031

Bits	Field Name	Default NVM Value	Description
15:0	PCI_LINKCAP H		

6.3.21.51 Address Low at PCI_GLBL_CNF - 0x0032

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_GLBL_CNF	0x110B8	
3:0	Type	0x1	

6.3.21.52 Address High at PCI_GLBL_CNF - 0x0033

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_GLBL_CNF		

6.3.21.53 PCI_GLBL_CNF L - 0x0034

Wake# enable

Bits	Field Name	Default NVM Value	Description
15:8	PCIE_CLKGATE_TIMER	0x10	Clock gating idle timer. This field defines the number of clocks (CSR clock) of idle-detect before gating the PCIE clocks.
7:6	RESERVED	0x0	
5	PCIE_PCLK_GATE_EN	0x0	When set to 1b, and if PCIE_CLKGATE_DIS is 0, the PCIE PCLK will also be dynamically gated.
4	PCIE_CLKGATE_L1_ONLY	0x1	When set to 1b, and if PCIE_CLKGATE_DIS is 0, the clock gating of the PCIE clocks will be only when the PCIE is in L1 state.

Bits	Field Name	Default NVM Value	Description
3	PCIE_CLKGAT_DIS	0x1	When set to 0b enables dynamic clock gating of the PCIE clocks (LCB, HIU & CSR).
2	WAKE_PIN_EN	0x0	When set to 1b enables the use of PE_WAKE_N pin for PME event in all power states, otherwise, it is enabled only in Dr state.
1:0	RESERVED	0x0	

6.3.21.54 PCI_GLBL_CNF H - 0x0035

Bits	Field Name	Default NVM Value	Description
15:3	RESERVED	0x0	
2	DIS_SCREAM_ON_D ATA_PARITY	0x1	
1	DIS_SCREAM_ON_C MD_PARITY	0x1	
0	DIS_STOP_ON_CMD _PARITY	0x1	

6.3.21.55 Address Low at PCI_VENDORID - 0x0036

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_VENDORID	0x1103C	
3:0	Type	0x1	

6.3.21.56 Address High at PCI_VENDORID - 0x0037

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_VENDORID		

6.3.21.57 PCI_VENDORID L - 0x0038

Wake# enable

Bits	Field Name	Default NVM Value	Description
15:0	PCI_VENDORID L	0x8086	

6.3.21.58 PCI_VENDORID H - 0x0039

Bits	Field Name	Default NVM Value	Description
15:0	PCI_VENDORID H	0x0	

6.3.21.59 Address Low at PCI_PFDEVID - 0x003A

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_PFDEVID	0x11008	
3:0	Type	0x1	

6.3.21.60 Address High at PCI_PFDEVID - 0x003B

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_PFDEVID		

6.3.21.61 PCI_PFDEVID Low (LAN) - 0x003C

Bits	Field Name	Default NVM Value	Description
15:0	PF_DEV_ID_LAN	0x57AC	If the Load Device ID in offset 0x7 of section PCIe General configuration is set, this word is loaded to the device ID of the LAN function.

6.3.21.62 PCI_PFDEVID High (SAN) - 0x003D

Bits	Field Name	Default NVM Value	Description
15:0	PF_DEV_ID_SAN	0x57AC	

6.3.21.63 Address Low at PCI_CNF - 0x003E

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_CNF	0x11000	
3:0	Type	0x1	

6.3.21.64 Address High at PCI_CNF - 0x003F

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_CNF		

6.3.21.65 PCI_CNF Low - 0x0040

Bits	Field Name	Default NVM Value	Description
15:7	Reserved	0x0	Reserved
6:5	INT_PIN	0x0	Controls the value advertised in the Interrupt Pin field of the PCI configuration header for this function. Values of 00b, 01b, 10b and 11b correspond to INTA#, INTB#, INTC# and INTD# respectively. The value advertised in the PCI configuration header is the value loaded from NVM + 1.
4	IO_BAR	0x0	I/O BAR support 0b - I/O BAR is not supported 1b - I/O BAR is supported
3	EXROM_DIS	0x0	Expansion ROM Disable 0b - The Expansion ROM BAR in the PCI configuration space is enabled 1b - The Expansion ROM BAR in the PCI configuration space is disabled
2:0	Reserved	0x0	Reserved

6.3.21.66 PCI_CNF High - 0x0041

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	Reserved

6.3.21.67 Address Low at PCI_VFDEVID - 0x0042

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_VFDEVID	0x11010	
3:0	Type	0x1	

6.3.21.68 Address High at PCI_VFDEVID - 0x0043

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_VFDEVID		

6.3.21.69 PCI_VFDEVID Low (LAN) - 0x0044

Bits	Field Name	Default NVM Value	Description
15:0	VF_DEV_ID_LAN	0x57AD	If the Load Device ID in offset 0x7 of section PCIe General configuration is set, this word is loaded to the device ID of the LAN function.

6.3.21.70 PCI_VFDEVID High (SAN) - 0x0045

Bits	Field Name	Default NVM Value	Description
15:0	VF_DEV_ID_SAN	0x57AD	

6.3.21.71 Address Low at PCI_CLASS - 0x0046

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_CLASS	0x11038	
3:0	Type	0x1	

6.3.21.72 Address High at PCI_CLASS - 0x0047

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_CLASS		

6.3.21.73 PCI_CLASS Low - 0x0048

Bits	Field Name	Default NVM Value	Description
15:1	Reserved	0x0	Reserved
0	STORAGE_CLASS	0x0	0b - The class code of this port is set to 0x020000 (LAN) 1b - The class code of this port is set to 0x010000 (SCSI)

6.3.21.74 PCI_CLASS High - 0x0049

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	Reserved

6.3.21.75 Address Low at PCI_FUNC2 - 0x004A

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_FUNC2	0x11018	
3:0	Type	0x1	

6.3.21.76 Address High at PCI_FUNC2 - 0x004B

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_FUNC2		

6.3.21.77 PCI_FUNC2 Low - 0x004C

Bits	Field Name	Default NVM Value	Description
15:1	Reserved	0x0	Reserved
0	FW_FUNC_DIS	0x0	FW Function Disable - Set by the FW when a CLP string or BMC requires that a function is disabled 0b - Function is enabled 1b - Function is disabled

6.3.21.78 PCI_FUNC2 High - 0x004D

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	Reserved

6.3.21.79 Address Low at PCI_PFDEVID - 0x004E

For inner structure please See [section 6.3.21.59](#)

6.3.21.80 Address High at PCI_PFDEVID - 0x004F

For inner structure please See [section 6.3.21.60](#)

6.3.21.81 PCI_PFDEVID Low (LAN) - 0x0050

For inner structure please See [section 6.3.21.61](#)

6.3.21.82 PCI_PFDEVID High (SAN) - 0x0051

For inner structure please See [section 6.3.21.62](#)

6.3.21.83 Address Low at PCI_CNF - 0x0052

For inner structure please See [section 6.3.21.63](#)

6.3.21.84 Address High at PCI_CNF - 0x0053

For inner structure please See [section 6.3.21.64](#)

6.3.21.85 PCI_CNF Low - 0x0054

For inner structure please See [section 6.3.21.65](#)

6.3.21.86 PCI_CNF High - 0x0055

For inner structure please See [section 6.3.21.66](#)

6.3.21.87 Address Low at PCI_VFDEVID - 0x0056

For inner structure please See [section 6.3.21.67](#)

6.3.21.88 Address High at PCI_VFDEVID - 0x0057

For inner structure please See [section 6.3.21.68](#)

6.3.21.89 PCI_VFDEVID Low (LAN) - 0x0058

For inner structure please See [section 6.3.21.69](#)

6.3.21.90 PCI_VFDEVID High (SAN) - 0x0059

For inner structure please See [section 6.3.21.70](#)

6.3.21.91 Address Low at PCI_CLASS - 0x005A

For inner structure please See [section 6.3.21.71](#)

6.3.21.92 Address High at PCI_CLASS - 0x005B

For inner structure please See [section 6.3.21.72](#)

6.3.21.93 PCI_CLASS Low - 0x005C

For inner structure please See [section 6.3.21.73](#)

6.3.21.94 PCI_CLASS High - 0x005D

For inner structure please See [section 6.3.21.74](#)

6.3.21.95 Address Low at PCI_FUNC2 - 0x005E

For inner structure please See [section 6.3.21.75](#)

6.3.21.96 Address High at PCI_FUNC2 - 0x005F

For inner structure please See [section 6.3.21.76](#)

6.3.21.97 PCI_FUNC2 Low - 0x0060

For inner structure please See [section 6.3.21.77](#)

6.3.21.98 PCI_FUNC2 High - 0x0061

For inner structure please See [section 6.3.21.78](#)

6.3.22 POR Registers PFA Autoload Module Section Summary Table

Lists registers addresses and values loaded at Power On Reset.

Word Offset	Description	Reference
0x0000	POR Registers Autoload Type	See section 6.3.22.1
0x0001	Module Length	See section 6.3.22.2
0x0002	Address Low at GL_MNG_HWARB_CTRL	See section 6.3.22.3
0x0003	Address High at GL_MNG_HWARB_CTRL	See section 6.3.22.4
0x0004	Data Low of GL_MNG_HWARB_CTRL	See section 6.3.22.5
0x0005	Data High of GL_MNG_HWARB_CTRL	See section 6.3.22.6
0x0006	Address Low at Port 0 PRTGEN_CNF	See section 6.3.22.7
0x0007	Address High at Port 0 PRTGEN_CNF	See section 6.3.22.8
0x0008	PRTGEN_CNF L	See section 6.3.22.9
0x0009	PRTGEN_CNF H	See section 6.3.22.10
0x000A	Address Low at Port 1 PRTGEN_CNF	See section 6.3.22.11
0x000B	Address High at Port 1 PRTGEN_CNF	See section 6.3.22.12
0x000C	PRTGEN_CNF L	See section 6.3.22.13
0x000D	PRTGEN_CNF H	See section 6.3.22.14
0x000E	Address Low at PF 0 PCI_FUNC	See section 6.3.22.15
0x000F	Address High at PF 0 PCI_FUNC	See section 6.3.22.16
0x0010	PCI_FUNC L	See section 6.3.22.17
0x0011	PCI_FUNC H	See section 6.3.22.18
0x0012	Address Low at PF 1 PCI_FUNC	See section 6.3.22.19
0x0013	Address High at PF 1 PCI_FUNC	See section 6.3.22.20
0x0014	PCI_FUNC L	See section 6.3.22.21
0x0015	PCI_FUNC H	See section 6.3.22.22
0x0016	Address Low at IOSFSB_LTR	See section 6.3.22.23
0x0017	Address High at IOSFSB_LTR	See section 6.3.22.24
0x0018	IOSFSB_LTR L	See section 6.3.22.25
0x0019	IOSFSB_LTR H	See section 6.3.22.26

6.3.22.1 POR Registers Autoload Type - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x118	Valid values are: 0x118 = POR Registers Autoload

6.3.22.2 Module Length - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 2 First Section -> Word: POR Registers PFA Autoload Module -> POR Registers Autoload Type Last Section -> Word: POR Registers PFA Autoload Module -> IOSFSB_LTR H

6.3.22.3 Address Low at GL_MNG_HWARB_CTRL - 0x0002

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GL_MNG_HWARB_CTRL	0xB6130	
3:0	Type	0x1	

6.3.22.4 Address High at GL_MNG_HWARB_CTRL - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GL_MNG_HWARB_CTRL		

6.3.22.5 Data Low of GL_MNG_HWARB_CTRL - 0x0004

Bits	Field Name	Default NVM Value	Description
15:1	RESERVED	0x0	
0	NCSI_ARB_EN	0x1	Hardware Arbitration Enable. If this bit is set, it is assumed the NCSI_ARB_IN and NCSI_ARB_OUT are connected to an Hardware arbitration ring. Otherwise, the NCSI_ARB_IN pin is pulled up internally.

6.3.22.6 Data High of GL_MNG_HWARB_CTRL - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	RESERVED cont		

6.3.22.7 Address Low at Port 0 PRTGEN_CNF - 0x0006

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTGEN_CNF	0x83120	
3:0	Type	0x1	

6.3.22.8 Address High at Port 0 PRTGEN_CNF - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTGEN_CNF		

6.3.22.9 PRTGEN_CNF L - 0x0008

Bits	Field Name	Default NVM Value	Description
15:3	RSVD	0x0	Reserved
2	EMP_PORT_DIS	0x0	Set by EMP to disable the Ethernet port. The NVM value for this bit should be 0b (enabled). NVM should use the PORT_DIS bit to disable a port
1	ALLOW_PORT_DIS	0x0	Allow Port Disable 0b = Asserting DEV_DIS_N has no effect on this port 1b = Asserting DEV_DIS_N disables this port
0	PORT_DIS	0x0	Port Disable. Defines if the Ethernet port is enabled from the NVM Exception: Port 0 is always enabled and cannot be disabled from the NVM 0b = Enabled 1b = Disabled

6.3.22.10 PRTGEN_CNF H - 0x0009

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	

6.3.22.11 Address Low at Port 1 PRTGEN_CNF - 0x000A

For inner structure please See [section 6.3.22.7](#)

6.3.22.12 Address High at Port 1 PRTGEN_CNF - 0x000B

For inner structure please See [section 6.3.22.8](#)

6.3.22.13 PRTGEN_CNF L - 0x000C

For inner structure please See [section 6.3.22.9](#)

6.3.22.14 PRTGEN_CNF H - 0x000D

For inner structure please See [section 6.3.22.10](#)

6.3.22.15 Address Low at PF 0 PCI_FUNC - 0x000E

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_FUNC	0x011020	
3:0	Type	0x1	

6.3.22.16 Address High at PF 0 PCI_FUNC - 0x000F

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_FUNC		

6.3.22.17 PCI_FUNC L - 0x0010

Bits	Field Name	Default NVM Value	Description
15:3	RSVD	0x0	Reserved
2	DIS_FUNC_ON_PORT_DIS	0x0	Defines whether this PF is disabled when the DEV_DIS_N pin is asserted. 0b = Asserting DEV_DIS_N has no effect on this PCI function. 1b = Asserting DEV_DIS_N disables this PCI function.
1	ALLOW_FUNC_DIS	0x0	Allow Function Disable. 0b = Asserting PCI_DIS_N has no effect on this PCI function. 1b = Asserting PCI_DIS_N disables this PCI function.
0	FUNC_DIS	0x0	Function Disable. Defines if the PCI function is enabled from the NVM. Exception: This bit is RO for PF0. It is always enabled and cannot be disabled from the NVM. 0b = Enabled. 1b = Disabled. Default: 0b for PF0. 1b for other functions.

6.3.22.18 PCI_FUNC H - 0x0011

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	

6.3.22.19 Address Low at PF 1 PCI_FUNC - 0x0012

For inner structure please See [section 6.3.22.15](#)

6.3.22.20 Address High at PF 1 PCI_FUNC - 0x0013

For inner structure please See [section 6.3.22.16](#)

6.3.22.21 PCI_FUNC L - 0x0014

For inner structure please See [section 6.3.22.17](#)

6.3.22.22 PCI_FUNC H - 0x0015

For inner structure please See [section 6.3.22.18](#)

6.3.22.23 Address Low at IOSFSB_LTR - 0x0016

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of IOSFSB_LTR	0xA4240	
3:0	Type	0x1	

6.3.22.24 Address High at IOSFSB_LTR - 0x0017

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of IOSFSB_LTR		

6.3.22.25 IOSFSB_LTR L - 0x0018

Bits	Field Name	Default NVM Value	Description
15:0	IOSFSB_LTR L	0x0040	

6.3.22.26 IOSFSB_LTR H - 0x0019

Bits	Field Name	Default NVM Value	Description
15:0	IOSFSB_LTR H	0x0	

6.3.23 MinSrev Section Summary Table

MinSrev TLV can be updated only if "Allow special update" bit is set in the command's flag

Word Offset	Description	Reference
0x0000	MinSrev Module Type	See section 6.3.23.1
0x0001	Length	See section 6.3.23.2
0x0002	Validity	See section 6.3.23.3
0x0003	NVM MinSrev LSB	See section 6.3.23.4
0x0004	NVM MinSrev MSB	See section 6.3.23.5
0x0005	OROM MinSrev LSB	See section 6.3.23.6
0x0006	OROM MinSrev MSB	See section 6.3.23.7

6.3.23.1 MinSrev Module Type - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x130	Valid values are: 0x130 = MinSrev Module

6.3.23.2 Length - 0x0001

Length of section

Bits	Field Name	Default NVM Value	Description
15:0	Block Length		Length in: 2 Bytes unit - 2 First Section -> Word: MinSrev -> MinSrev Module Type Last Section -> Word: MinSrev -> OROM MinSrev MSB Section length in words Note: There are 21 MAC addresses to support switch mode.

6.3.23.3 Validity - 0x0002

Validity of MinSrev words

Bits	Field Name	Default NVM Value	Description
15:2	Reserved	0x0000	
1	OROM	0x0	OROM MinSrev valid
0	NVM	0x0	NVM MinSrev valid

6.3.23.4 NVM MinSrev LSB - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	MinSrev LSB	0x0000	NVM MinSrev

6.3.23.5 NVM MinSrev MSB - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	MinSrev MSB	0x0000	NVM MinSrev

6.3.23.6 OROM MinSrev LSB - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	MinSrev LSB	0x0000	NVM MinSrev

6.3.23.7 OROM MinSrev MSB - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	MinSrev MSB	0x0000	NVM MinSrev

6.3.24 PF MAC Address Section Summary Table

PF/port MAC addresses PFA section

Word Offset	Description	Reference
0x0000	PCI Serial id MAC Address Module Type	See section 6.3.24.1
0x0001	Length	See section 6.3.24.2
0x0002	PCI_SERL0	See section 6.3.24.3
0x0003	PCI_SERL1	See section 6.3.24.4
0x0004	PCI_SERH0	See section 6.3.24.5
0x0005	Reserved	See section 6.3.24.6
0x0006	PF MAC Address Module Type	See section 6.3.24.7
0x0007	Section Header	See section 6.3.24.8
0x0008 + 4*n, n=0...1	PFPM_SAL0	See section 6.3.24.9
0x0009 + 4*n, n=0...1	PFPM_SAL1	See section 6.3.24.10
0x000A + 4*n, n=0...1	PFPM_SAH0	See section 6.3.24.11
0x000B + 4*n, n=0...1	PFPM_SAH1	See section 6.3.24.12

6.3.24.1 PCI Serial id MAC Address Module Type - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x133	Valid values are: 0x133 = PCI Serial id MAC addresses Module

6.3.24.2 Length - 0x0001

Length of MAC address section

Bits	Field Name	Default NVM Value	Description
15:0	Block Length		Length in: 2 Bytes unit - 2 First Section -> Word: PF MAC Address -> PCI Serial id MAC Address Module Type Last Section -> Word: PF MAC Address -> PFPM_SAH1

6.3.24.3 PCI_SERL0 - 0x0002

This word is loaded by FW to PCI_SERL 0:15

Bits	Field Name	Default NVM Value	Description
15:0	PCI_SERL0	0x0000	See respective bits in the PCI_SERL register.

6.3.24.4 PCI_SERL1 - 0x0003

This word is loaded by FW to PCI_SERL 16:31

Bits	Field Name	Default NVM Value	Description
15:0	PCI_SERL1	0x0000	See respective bits in the PCI_SERL register.

6.3.24.5 PCI_SERH0 - 0x0004

This word is loaded by FW to PCI_SERH 0:15

Bits	Field Name	Default NVM Value	Description
15:0	PCI_SERH0	0x0001	See respective bits in the PCI_SERH register.

6.3.24.6 Reserved - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0000	

6.3.24.7 PF MAC Address Module Type - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x10F	Valid values are: 0x10F = PF MAC addresses Module

6.3.24.8 Section Header - 0x0007

Length of MAC address section

Bits	Field Name	Default NVM Value	Description
15:0	Block Length		Length in: 2 Bytes unit - 2 First Section -> Word: PF MAC Address -> PF MAC Address Module Type Last Section -> Word: PF MAC Address -> PFPD_SAH1 Section length in words Note: There are 21 MAC addresses to support switch mode.

6.3.24.9 PFPM_SAL0[n] (0x0008 + 4*n, n=0...1)

This word is loaded by FW to the appropriate RAL register, bytes [1,0].

One MAC address per enabled PF/Port

This is a per device per enabled PF value allocated by manufacturing.

Bits	Field Name	Default NVM Value	Description
15:0	PFPM_SAL0	0x0000	See respective bits in the RAL register.

6.3.24.10 PFPM_SAL1[n] (0x0009 + 4*n, n=0...1)

This register is loaded by FW to the appropriate RAL register, bytes [3:2].

One MAC address per enabled PF.

This is a per device per enabled PF value allocated by manufacturing.

Bits	Field Name	Default NVM Value	Description
15:0	PFPM_SAL1	0x0000	See respective bits in the RAL register.

6.3.24.11 PFPM_SAH0[n] (0x000A + 4*n, n=0...1)

This register is loaded by FW to the appropriate RAH register, bytes [1,0].

One MAC address per enabled PF/Port

This is a per device per enabled PF value allocated by manufacturing.

Bits	Field Name	Default NVM Value	Description
15:0	PFPM_SAH0	0x0000	See respective bits in the RAH register.

6.3.24.12 PFPM_SAH1[n] (0x000B + 4*n, n=0...1)

This register is loaded by FW to the appropriate RAH register, bytes [3:2].

One MAC address per enabled PF/Port

This is a per device per enabled PF value allocated by manufacturing.

Bits	Field Name	Default NVM Value	Description
15:0	PFPM_SAH1	0x0000	See respective bits in the RAH register.

6.3.25 MNG MAC Address Section Summary Table

Manageability MAC addresses PFA section

Word Offset	Description	Reference
0x0000	MNG MAC Address Module Type	See section 6.3.25.1
0x0001	Section Header - Length	See section 6.3.25.2
0x0002 + 3*n, n=0...31	LAN Ethernet MAC Address (LSB) MMAL	See section 6.3.25.3
0x0003 + 3*n, n=0...31	LAN Ethernet MAC Address (Mid) MMAL	See section 6.3.25.4
0x0004 + 3*n, n=0...31	LAN Ethernet MAC Address (MSB) MMAH	See section 6.3.25.5

6.3.25.1 MNG MAC Address Module Type - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x110	Valid values are: 0x110 = MNG MAC addresses Module

6.3.25.2 Section Header - Length - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Block Length		Length in: 2 Bytes unit - 2 First Section -> Word: MNG MAC Address -> MNG MAC Address Module Type Last Section -> Word: MNG MAC Address -> LAN Ethernet MAC Address (MSB) MMAH Section length in words

6.3.25.3 LAN Ethernet MAC Address (LSB) MMAL[n] (0x0002 + 3*n, n=0...31)

This word is loaded by the FW to the 16 LS bits of the MMAL[0-3] register of port n.

The index equals 4*Port# + MAC#

It is used as the MAC address when dedicated MAC address mode is used in legacy SMBus.

This is a per device per port value allocated by manufacturing.

Bits	Field Name	Default NVM Value	Description
15:8	Ethernet MAC Address, Byte 1	0xFF	
7:0	Ethernet MAC Address, Byte 0	0xFF	

6.3.25.4 LAN Ethernet MAC Address (Mid) MMAL[n] (0x0003 + 3*n, n=0...31)

This word is loaded by the FW to the 16 MS bits of the MMAL[0-3] register of port n.

The index equals 4*Port# + MAC#

It is used as the MAC address when dedicated MAC address mode is used in legacy SMBus.

This is a per device per port value allocated by manufacturing.

Bits	Field Name	Default NVM Value	Description
15:8	Ethernet MAC Address, Byte 3	0xFF	
7:0	Ethernet MAC Address, Byte 2	0xFF	

6.3.25.5 LAN Ethernet MAC Address (MSB) MMAH[n] (0x0004 + 3*n, n=0...31)

This word is loaded by the FW to the MMAH[0-3] register of port n.

The index equals 4*Port# + MAC#

It is used as the MAC address when dedicated MAC address mode is used in legacy SMBus.

This is a per device per port value allocated by manufacturing.

Bits	Field Name	Default NVM Value	Description
15:8	Ethernet MAC Address, Byte 5	0xff	
7:0	Ethernet MAC Address, Byte 4	0xff	

6.3.26 FW Logging Defaults Section Summary Table

UART debug defaults

Word Offset	Description	Reference
0x0000	Sub Module Type	See section 6.3.26.1
0x0001	Section Length	See section 6.3.26.2
0x0002	UART Control	See section 6.3.26.3
0x0003	Module Logging Enable [n=0]	See section 6.3.26.4
0x0004	Module Logging Enable [n=1]	See section 6.3.26.5
0x0005	Module Logging Enable [n=2]	See section 6.3.26.6
0x0006	Module Logging Enable [n=3]	See section 6.3.26.7
0x0007	Module Logging Enable [n=4]	See section 6.3.26.8
0x0008	Module Logging Enable [n=5]	See section 6.3.26.9
0x0009	Module Logging Enable [n=6]	See section 6.3.26.10
0x000A	Module Logging Enable [n=7]	See section 6.3.26.11

6.3.26.1 Sub Module Type - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x11D	Valid values are: 0x11D = UART Debug

6.3.26.2 Section Length - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Section Length		Length in: 2 Bytes unit - 2 First Section -> Word: FW Logging Defaults -> Sub Module Type Last Section -> Word: FW Logging Defaults -> Module Logging Enable [n=7] Section length in words

6.3.26.3 UART Control - 0x0002

Bits	Field Name	Default NVM Value	Description
15:1	Reserved	0x0	
0	UART Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled

6.3.26.4 Module Logging Enable [n=0] - 0x0003

Bits	Field Name	Default NVM Value	Description
15	Module Link Topology Detection Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
14	Module Link Topology Detection Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
13	Module Link Topology Detection Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
12	Module Link Topology Detection Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
11	Module Link Management Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
10	Module Link Management Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
9	Module Link Management Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
8	Module Link Management Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
7	Module Control Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
6	Module Control Warning Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
5	Module Control Normal Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
4	Module Control Verbose Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
3	Module General Error Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
2	Module General Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
1	Module General Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
0	Module General Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled

6.3.26.5 Module Logging Enable [n=1] - 0x0004

Bits	Field Name	Default NVM Value	Description
15	Module MDIO Error Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
14	Module MDIO Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
13	Module MDIO Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
12	Module MDIO Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
11	Module SDP Error Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
10	Module SDP Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
9	Module SDP Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
8	Module SDP Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
7	Module I2C Error Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
6	Module I2C Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
5	Module I2C Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
4	Module I2C Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
3	Module Dreadnought Lake Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
2	Module Dreadnought Lake Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
1	Module Dreadnought Lake Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
0	Module Dreadnought Lake Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled

6.3.26.6 Module Logging Enable [n=2] - 0x0005

Bits	Field Name	Default NVM Value	Description
15	Module DCBx Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
14	Module DCBx Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
13	Module DCBx Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
12	Module DCBx Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
11	Module LLDP Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
10	Module LLDP Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
9	Module LLDP Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
8	Module LLDP Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
7	Module HDMA Error Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
6	Module HDMA Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
5	Module HDMA Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
4	Module HDMA Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
3	Module Admin Queue Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
2	Module Admin Queue Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
1	Module Admin Queue Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
0	Module Admin Queue Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled

6.3.26.7 Module Logging Enable [n=3] - 0x0006

Bits	Field Name	Default NVM Value	Description
15	Module Authentication Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
14	Module Authentication Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
13	Module Authentication Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
12	Module Authentication Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
11	Module NVM Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
10	Module NVM Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
9	Module NVM Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
8	Module NVM Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
7	Module NetProxy -> XLR Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
6	Module NetProxy -> XLR Warning Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
5	Module NetProxy -> XLR Normal Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
4	Module NetProxy -> XLR Verbose Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
3	Module DCB Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
2	Module DCB Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
1	Module DCB Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
0	Module DCB Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled

6.3.26.8 Module Logging Enable [n=4] - 0x0007

Bits	Field Name	Default NVM Value	Description
15	Module Switch Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
14	Module Switch Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
13	Module Switch Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
12	Module Switch Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
11	Module Parser Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
10	Module Parser Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
9	Module Parser Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
8	Module Parser Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
7	Module IOSF Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
6	Module IOSF Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
5	Module IOSF Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
4	Module IOSF Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
3	Module VPD Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
2	Module VPD Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
1	Module VPD Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
0	Module VPD Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled

6.3.26.9 Module Logging Enable [n=5] - 0x0008

Bits	Field Name	Default NVM Value	Description
15	Module Post Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
14	Module Post Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
13	Module Post Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
12	Module Post Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
11	Module ACL Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
10	Module ACL Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
9	Module ACL Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
8	Module ACL Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
7	Module TX Queue Management Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
6	Module TX Queue Management Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
5	Module TX Queue Management Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
4	Module TX Queue Management Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
3	Module Scheduler Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
2	Module Scheduler Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
1	Module Scheduler Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
0	Module Scheduler Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled

6.3.26.10 Module Logging Enable [n=6] - 0x0009

Bits	Field Name	Default NVM Value	Description
15	Module SyncE Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
14	Module SyncE Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
13	Module SyncE Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
12	Module SyncE Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
11	Module Manageability Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
10	Module Manageability Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
9	Module Manageability Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
8	Module Manageability Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
7	Module Task Dispatcher Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
6	Module Task Dispatcher Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
5	Module Task Dispatcher Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
4	Module Task Dispatcher Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
3	Module Watchdog Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
2	Module Watchdog Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
1	Module Watchdog Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
0	Module Watchdog Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled

6.3.26.11 Module Logging Enable [n=7] - 0x000A

Bits	Field Name	Default NVM Value	Description
15	Module Version Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
14	Module Version Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
13	Module Version Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
12	Module Version Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
11	Module PF Registration Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
10	Module PF Registration Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
9	Module PF Registration Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
8	Module PF Registration Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
7	Module Time Sync Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
6	Module Time Sync Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
5	Module Time Sync Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
4	Module Time Sync Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
3	Module Health Error Logging Enable	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
2	Module Health Warning Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
1	Module Health Normal Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
0	Module Health Verbose Logging Enable	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled

6.3.27 MD Link Topology Section Summary Table

This is the Ethernet Link topology description of the motherboard. Link topology is described in EAS section 3.5

Word Offset	Description	Reference
0x0000	FW MNG Link Topology Module Type	See section 6.3.27.1
0x0001	FW MNG Link Topology Module Length	See section 6.3.27.2
0x0002	Generic info	See section 6.3.27.3
0x0003	Netlist Version	See section 6.3.27.4
0x0004 + 1*n, n=0...15	Pair PHY Type	See section 6.3.27.5
0x0014	Port bitmap 0	See section 6.3.27.6
0x0015	Port bitmap 1	See section 6.3.27.7

6.3.27.1 FW MNG Link Topology Module Type - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x120	Valid values are: 0x120 = FW mng Link Topology

6.3.27.2 FW MNG Link Topology Module Length - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Block Length		Length in: 2 Bytes unit - 2 First Section -> Word: MD Link Topology -> FW MNG Link Topology Module Type Last Section -> Word: MD Link Topology -> Port bitmap 1 Section length in words

6.3.27.3 Generic info - 0x0002

Bits	Field Name	Default NVM Value	Description
15:8	Netlist Map Version	0x1	The version of the Net-list field definition.
7:1	Reserved	0x0	
0	Load Mode	0x1	Net-list Loading Mode - Controls the steps taken at net-list load time. Valid only for the motherboard's net-list. There are two modes: Normal Mode - The active Port Options are loaded. Resolution Mode - The active Port Options of non-innermost PHY nodes is updated and then the active Port Options are loaded. The update is performed based on matching the active Port Option of the connected Innermost PHY. If the Port Option was forced the active option is not updated. Valid values are: 0x0 = Normal Mode 0x1 = Resolution Mode

6.3.27.4 Netlist Version - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	Netlist Version	0x0	The version of the Net-list.

6.3.27.5 Pair PHY Type[n] (0x0004 + 1*n, n=0...15)

Bits	Field Name	Default NVM Value	Description
15:12	PHY 2*n+1 Reserved	0x0	
11:8	PHY 2*n+1 active option	0x0	The PHY active port option.
7:4	PHY 2*n Reserved	0x0	
3:0	PHY 2*n active option	0x0	The PHY active port option.

6.3.27.6 Port bitmap 0 - 0x0014

Bits	Field Name	Default NVM Value	Description
15:0	Port bitmap 0	0x0	

6.3.27.7 Port bitmap 1 - 0x0015

Bits	Field Name	Default NVM Value	Description
15:4	Reserved	0x0	
3:0	Port bitmap 1	0x0	

6.3.28 RDE Module Section Summary Table

LLDP Preserved

Word Offset	Description	Reference
0x0000	Type	See section 6.3.28.1
0x0001	Length	See section 6.3.28.2
0x0002	AssetTag	See section 6.3.28.3
0x0003	AssetTag0	See section 6.3.28.4
0x0004	AssetTag1	See section 6.3.28.5
0x0005	AssetTag2	See section 6.3.28.6
0x0006	AssetTag3	See section 6.3.28.7
0x0007	AssetTag4	See section 6.3.28.8
0x0008	AssetTag5	See section 6.3.28.9
0x0009	AssetTag6	See section 6.3.28.10
0x000A	AssetTag7	See section 6.3.28.11
0x000B	AssetTag8	See section 6.3.28.12
0x000C	AssetTag9	See section 6.3.28.13
0x000D	AssetTag10	See section 6.3.28.14
0x000E	AssetTag11	See section 6.3.28.15
0x000F	AssetTag12	See section 6.3.28.16
0x0010	AssetTag13	See section 6.3.28.17
0x0011	AssetTag14	See section 6.3.28.18
0x0012	AssetTag15	See section 6.3.28.19
0x0013	AssetTag16	See section 6.3.28.20
0x0014	AssetTag17	See section 6.3.28.21
0x0015	AssetTag18	See section 6.3.28.22
0x0016	AssetTag19	See section 6.3.28.23
0x0017	AssetTag20	See section 6.3.28.24
0x0018	AssetTag21	See section 6.3.28.25
0x0019	AssetTag22	See section 6.3.28.26
0x001A	AssetTag23	See section 6.3.28.27
0x001B	AssetTag24	See section 6.3.28.28
0x001C	AssetTag25	See section 6.3.28.29
0x001D	AssetTag26	See section 6.3.28.30
0x001E	AssetTag27	See section 6.3.28.31
0x001F	AssetTag28	See section 6.3.28.32
0x0020	AssetTag29	See section 6.3.28.33
0x0021	AssetTag30	See section 6.3.28.34

6.3.28.1 Type - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x131	Valid values are: 0x131 = RDE module

6.3.28.2 Length - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit - 2 First Section -> Word: RDE Module -> Type Last Section -> Word: RDE Module -> AssetTag30 Section length in words

6.3.28.3 AssetTag - 0x0002

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.4 AssetTag0 - 0x0003

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.5 AssetTag1 - 0x0004

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.6 AssetTag2 - 0x0005

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.7 AssetTag3 - 0x0006

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.8 AssetTag4 - 0x0007

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.9 AssetTag5 - 0x0008

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.10 AssetTag6 - 0x0009

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.11 AssetTag7 - 0x000A

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.12 AssetTag8 - 0x000B

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.13 AssetTag9 - 0x000C

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.14 AssetTag10 - 0x000D

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.15 AssetTag11 - 0x000E

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.16 AssetTag12 - 0x000F

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.17 AssetTag13 - 0x0010

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.18 AssetTag14 - 0x0011

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.19 AssetTag15 - 0x0012

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.20 AssetTag16 - 0x0013

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.21 AssetTag17 - 0x0014

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.22 AssetTag18 - 0x0015

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.23 AssetTag19 - 0x0016

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.24 AssetTag20 - 0x0017

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.25 AssetTag21 - 0x0018

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.26 AssetTag22 - 0x0019

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.27 AssetTag23 - 0x001A

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.28 AssetTag24 - 0x001B

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.29 AssetTag25 - 0x001C

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.30 AssetTag26 - 0x001D

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.31 AssetTag27 - 0x001E

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.32 AssetTag28 - 0x001F

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.33 AssetTag29 - 0x0020

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.28.34 AssetTag30 - 0x0021

Defines status of LLDP agent. Each LAN port has independent status.

- 3: Both receive and transmit enabled
- 2: LLDP is configured for transmits only
- 1: LLDP is configured for receives only
- 0: LLDP agent is disabled

Bits	Field Name	Default NVM Value	Description
15:0	AssetTag	0x0	Defines status of LLDP agent. Applies to LAN Port 0.

6.3.29 Identical content as PLDM header ComponentImageSetVersionString Section Summary Table

VPD area - Vital Product Data loaded by OEM. It contains RO and RW info about the NIC/LOM.

Word Offset	Description	Reference
0x0000	Sub Module Type	See section 6.3.29.1
0x0001	Length	See section 6.3.29.2
0x0002	ComponentImageSetVersionString - FW component prefix	See section 6.3.29.3
0x0003	ComponentImageSetVersionString - FW component version 0	See section 6.3.29.4
0x0004	ComponentImageSetVersionString - FW component version 1	See section 6.3.29.5
0x0005	ComponentImageSetVersionString - FW component version 2	See section 6.3.29.6
0x0006	ComponentImageSetVersionString - FW component version 3	See section 6.3.29.7
0x0007	ComponentImageSetVersionString - OROM component prefix	See section 6.3.29.8
0x0008	ComponentImageSetVersionString - OROM component version 0	See section 6.3.29.9
0x0009	ComponentImageSetVersionString - OROM component version 1	See section 6.3.29.10
0x000A	ComponentImageSetVersionString - OROM component version 2	See section 6.3.29.11
0x000B	ComponentImageSetVersionString - OROM component version 3	See section 6.3.29.12
0x000C	ComponentImageSetVersionString - Netlist component prefix	See section 6.3.29.13
0x000D	ComponentImageSetVersionString - Netlist component version 0	See section 6.3.29.14
0x000E	ComponentImageSetVersionString - Netlist component version 1	See section 6.3.29.15
0x000F	ComponentImageSetVersionString - Netlist component version 2	See section 6.3.29.16
0x0010	ComponentImageSetVersionString - Netlist component version 3	See section 6.3.29.17
0x0011	ComponentImageSetVersionString - null padding	See section 6.3.29.18

6.3.29.1 Sub Module Type - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x127	Valid values are: 0x127 = PLDM Component Image Set Version String

6.3.29.2 Length - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit - 2 First Section -> Word: Identical content as PLDM header ComponentImageSetVersionString -> Sub Module Type Last Section -> Word: Identical content as PLDM header ComponentImageSetVersionString -> ComponentImageSetVersionString - null padding

6.3.29.3 ComponentImageSetVersionString - FW component prefix - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - FW component prefix	0x3A4E	"N:" in ASCII

6.3.29.4 ComponentImageSetVersionString - FW component version 0 - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - FW component version 0	0x3030	FW component version 0-3: ASCII presentation of FW comparison stamp. Values are zero (in ASCII) padded.

6.3.29.5 ComponentImageSetVersionString - FW component version 1 - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - FW component version 1	0x3030	

6.3.29.6 ComponentImageSetVersionString - FW component version 2 - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - FW component version 2	0x3030	

6.3.29.7 ComponentImageSetVersionString - FW component version 3 - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - FW component version 3	0x3030	

6.3.29.8 ComponentImageSetVersionString - OROM component prefix - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - OROM component prefix	0x3A4F	"O:" in ASCII

6.3.29.9 ComponentImageSetVersionString - OROM component version 0 - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - OROM component version 0	0x3030	OROM component version 0-3: ASCII presentation of OROM comparison stamp. Values are zero (in ASCII) padded.

6.3.29.10 ComponentImageSetVersionString - OROM component version 1 - 0x0009

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - OROM component version 1	0x3030	

6.3.29.11 ComponentImageSetVersionString - OROM component version 2 - 0x000A

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - OROM component version 2	0x3030	

6.3.29.12 ComponentImageSetVersionString - OROM component version 3 - 0x000B

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - OROM component version 3	0x3030	

6.3.29.13 ComponentImageSetVersionString - Netlist component prefix - 0x000C

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - Netlist component prefix	0x3A54	"T:" in ASCII

6.3.29.14 ComponentImageSetVersionString - Netlist component version 0 - 0x000D

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - Netlist component version 0	0x3030	Netlist componenet version 0-3: ASCII presentation of Netlist comparison stamp. Values are zero (in ASCII) padded.

6.3.29.15 ComponentImageSetVersionString - Netlist component version 1 - 0x000E

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - Netlist component version 1	0x3030	

6.3.29.16 ComponentImageSetVersionString - Netlist component version 2 - 0x000F

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - Netlist component version 2	0x3030	

6.3.29.17 ComponentImageSetVersionString - Netlist component version 3 - 0x0010

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - Netlist component version 3	0x3030	

6.3.29.18 ComponentImageSetVersionString - null padding - 0x0011

Bits	Field Name	Default NVM Value	Description
15:0	ComponentImageSetVersionString - null padding	0x0	

6.3.30 Software Checksum Module Section Summary Table

The SW Checksum field covers the PFA portion of shadow RAM contents (reserved words included), Its value is computed such that after adding all the covered words, including the SW Checksum word itself, the sum is 0xBABA

The checksum word is used to ensure that the base NVM image is a valid image. The initial value in the 16-bit summing register should be 0x0000 and the carry bit should be ignored after each addition.

This word is verified and recalculated by Firmware upon Checksum AQC

Word Offset	Description	Reference
0x0000	Sub Module Type checksum	See section 6.3.30.1
0x0001	Checksum Module Length	See section 6.3.30.2
0x0002	Checksum	See section 6.3.30.3

6.3.30.1 Sub Module Type checksum - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x3F	Valid values are: 0x3F = Checksum

6.3.30.2 Checksum Module Length - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit - 2 First Section -> Word: Software Checksum Module -> Sub Module Type checksum Last Section -> Word: Software Checksum Module -> Checksum

6.3.30.3 Checksum - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	Checksum		

6.3.31 Link Default Override Mask Section Summary Table

Word Offset	Description	Reference
0x0000	Link Default Override Mask Type	See section 6.3.31.1
0x0001	Length	See section 6.3.31.2
0x0002 + 10*n, n=0...1	Port Options 0	See section 6.3.31.3
0x0003 + 10*n, n=0...1	Port Options 1	See section 6.3.31.4
0x0004 + 10*n, n=0...1	Port PHY types 0	See section 6.3.31.5
0x0005 + 10*n, n=0...1	Port PHY types 1	See section 6.3.31.6
0x0006 + 10*n, n=0...1	Port PHY types 2	See section 6.3.31.7
0x0007 + 10*n, n=0...1	Port PHY types 3	See section 6.3.31.8
0x0008 + 10*n, n=0...1	Port PHY types 4	See section 6.3.31.9
0x0009 + 10*n, n=0...1	Port PHY types 5	See section 6.3.31.10
0x000A + 10*n, n=0...1	Port PHY types 6	See section 6.3.31.11
0x000B + 10*n, n=0...1	Port PHY types 7	See section 6.3.31.12

6.3.31.1 Link Default Override Mask Type - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x134	Valid values are: 0x134 = Link Default Override Mask

6.3.31.2 Length - 0x0001

Length of section

Bits	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit - 2 First Section -> Word: Link Default Override Mask -> Link Default Override Mask Type Last Section -> Word: Link Default Override Mask -> Port PHY types 7 words

6.3.31.3 Port Options 0[n] (0x0002 + 10*n, n=0...1)

Port link options

Bits	Field Name	Default NVM Value	Description
15	Auto FEC enable	0x1	When "override enable" bit is set, the "Auto FEC enable" bit is taken from here and not from the topology net-list.
14	LESM enable	0x1	When "override enable" bit is set, the "LESM enable" bit is taken from here and not from the topology net-list.
13	RS_272_REQUEST	0x0	Controls RS FEC 272 capability request for 50G KR/CR, 100G KR2/CR2 and 200G KR4/CR4. 0b = FEC Disabled 1b = FEC Enabled (requested)
12	RS_272_200G_ABILITY	0x1	Controls RS FEC ability advertisement for 200G KR4/CR4. 0b = FEC Disabled 1b = FEC Enabled (advertised)
11	RS_272_100G_ABILITY	0x1	Controls RS FEC ability advertisement for 100G KR2/CR2. 0b = FEC Disabled 1b = FEC Enabled (advertised)
10	RS_272_50G_ABILITY	0x1	Controls RS FEC ability advertisement for 50G KR/CR. 0b = FEC Disabled 1b = FEC Enabled (advertised)
9:8	PAUSE ability	0x0	When "override enable" bit is set, the "PAUSE ability" bits are taken from here and not from the topology net-list.
7:6	Reserved	0x3	
5	EEE enable override	0x0	When "override enable" bit is set, the EEE can be enabled or disabled according to this bit, assuming that the PHY_TYPE is supporting EEE. 0 - Disable 1 - Enable
4	Disable automatic Link on Startup override	0x0	When "override enable" bit is set, the "Disable automatic link on Startup" feature is taken from here and not from the link topology. 0 - "Automatic Link on startup" feature is enabled 1 - "Automatic Link on startup" feature is disabled
3	Override Enable	0x0	Enables the override ability from PFA for this port. When this bit is set for a specific port, the parameters of this port are taken from the PFA and not from the link topology structures. 0 - Disable 1 - Enable
2	Port Disable behavior mode	0x0	Controls how to disable the port when requested. 0 - will disable only the host 1 - Disables all usages
1	EPCT ability to change Lenient/Strict enable	0x0	EPCT ability to change Lenient / Strict mode
0	Lenient/ Strict mode	0x0	Lenient = 0 Strict = 1

6.3.31.4 Port Options 1[n] (0x0003 + 10*n, n=0...1)

Port PHY options

Bits	Field Name	Default NVM Value	Description
15:14	Reserved	0x3	
13	Override FEC Setting	0x0	
12	Override LESM Enable	0x0	
11	Override PAUSE Setting	0x0	
10	Override EEE Setting	0x0	
9	Override "Disable automatic Link on Startup"	0x0	
8	Override PHY Types	0x0	
7	Reserved (FIRE_CODE_25_ABILITY)	0x1	Reserved (Controls KR FEC ability advertisement for 25G KR1/CR1. 0b = FEC Disabled 1b = FEC Enabled (advertised))
6	Reserved (RS_528_ABILITY)	0x1	Reserved (Controls RS FEC ability advertisement for 25G KR1/CR1. 0b = FEC Disabled 1b = FEC Enabled (advertised))
5	Reserved	0x0	
4	Reserved (RS_544_REQUEST)	0x0	Reserved (Controls RS FEC 544 capability request for 25G/50G lanes KR/KR-S/KR1/CR/CR-S/CR1.)
3	Reserved (FIRE_CODE_25_REQUEST)	0x1	Reserved (Controls KR FEC capability request for 25G lanes KR/KR-S/KR1/CR/CR-S/CR1. 0b = FEC Disabled 1b = FEC Enabled (requested))
2	Reserved (RS_528_REQUEST)	0x1	Reserved (Controls RS FEC 528 capability request for 25G lanes KR/KR-S/KR1/CR/CR-S/CR1. 0b = FEC Disabled 1b = FEC Enabled (requested))
1	FIRE_CODE_10_REQUEST	0x0	Controls FEC capability request for 10G KR 0b = FEC Disabled 1b = FEC Enabled (requested)
0	FIRE_CODE_10_ABILITY enable	0x0	Controls FEC capability advertisement for 10G KR 0b = FEC Disabled 1b = FEC Enabled (advertised)

6.3.31.5 Port PHY types 0 [n] (0x0004 + 10*n, n=0...1)

8 Words per port.

(to be updated)

Bits	Field Name	Default NVM Value	Description
15	10GBASE-LR	0x1	
14	10GBASE-SR	0x1	
13	10G-SFI-DA	0x1	
12	10GBASE-T	0x1	
11	5GBASE-KR	0x1	
10	5GBASE-T	0x1	
9	2.5GBASE-KX	0x1	
8	2.5GBASE-X	0x1	
7	2.5GBASE-T	0x1	
6	1G-SGMII	0x1	
5	1000BASE-KX	0x1	
4	1000BASE-LX	0x1	
3	1000BASE-SX	0x1	
2	1000BASE-T	0x1	
1	100M-SGMII	0x1	
0	100BASE-TX	0x1	

6.3.31.6 Port PHY types 1[n] (0x0005 + 10*n, n=0...1)

Bits	Field Name	Default NVM Value	Description
15	Reserved	0x1	
14	Reserved	0x1	
13	Reserved (25G-AUI-C2C)	0x1	
12	Reserved (25G-AUI-AOC\ACC)	0x1	
11	Reserved (25GBASE-KR1)	0x1	
10	Reserved (25GBASE-KR-S)	0x1	
9	Reserved (25GBASE-KR)	0x1	
8	Reserved (25GBASE-LR)	0x1	
7	Reserved (25GBASE-SR)	0x1	
6	Reserved (25GBASE-CR1)	0x1	
5	Reserved (25GBASE-CR-S)	0x1	
4	Reserved (25GBASE-CR)	0x1	
3	Reserved (25GBASE-T)	0x1	
2	10G-SFI-C2C	0x1	
1	10G-SFI-AOC\ACC	0x1	
0	10GBASE-KR\CR1	0x1	

6.3.31.7 Port PHY types 2[n] (0x0006 + 10*n, n=0...1)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	

6.3.31.8 Port PHY types 3[n] (0x0007 + 10*n, n=0...1)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	

6.3.31.9 Port PHY types 4[n] (0x0008 + 10*n, n=0...1)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	

6.3.31.10 Port PHY types 5[n] (0x0009 + 10*n, n=0...1)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	

6.3.31.11 Port PHY types 6[n] (0x000A + 10*n, n=0...1)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	

6.3.31.12 Port PHY types 7[n] (0x000B + 10*n, n=0...1)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	

6.3.32 RDE Ethernet MTU Section Summary Table

Word Offset	Description	Reference
0x0000	RDE Ethernet MTU Type	See section 6.3.32.1
0x0001	Length	See section 6.3.32.2
0x0002 + 1*n, n=0...7	Ethernet MTU size	See section 6.3.32.3

6.3.32.1 RDE Ethernet MTU Type - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x136	Valid values are: 0x136 = RDE Ethernet MTU Type

6.3.32.2 Length - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit - 2 First Section -> Word: RDE Ethernet MTU -> RDE Ethernet MTU Type Last Section -> Word: RDE Ethernet MTU -> Ethernet MTU size

6.3.32.3 Ethernet MTU size[n] (0x0002 + 1*n, n=0...7)

Bits	Field Name	Default NVM Value	Description
15:0	Ethernet MTU size	0xFFFF	

6.3.33 HII Port Disable By Function Section Summary Table

Word Offset	Description	Reference
0x0000	Type	See section 6.3.33.1
0x0001	Length	See section 6.3.33.2
0x0002	HII port disable by function	See section 6.3.33.3

6.3.33.1 Type - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x139	Valid values are: 0x139 = HII Port Disable By Function

6.3.33.2 Length - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit - 2 First Section -> Word: HII Port Disable By Function -> Type Last Section -> Word: HII Port Disable By Function -> HII port disable by function

6.3.33.3 HII port disable by function - 0x0002

Bits	Field Name	Default NVM Value	Description
15:2	Reserved	0x0	Reserved
1	HII port disable by function 1	0x0	When the bit is set, the firmware should disable the port that is associated with the PF according to PFGEN_PORTNUM Valid values are: 0x0 = Port Enabled 0x1 = Port Disabled
0	HII port disable by function 0	0x0	When the bit is set, the firmware should disable the port that is associated with the PF according to PFGEN_PORTNUM Valid values are: 0x0 = Port Enabled 0x1 = Port Disabled

6.3.34 NetlistMinSrev Section Summary Table

NetlistMinSrev TLV can be updated only if "Allow special update" bit is set in the command's flag

Word Offset	Description	Reference
0x0000	NetlistMinSrev Module Type	See section 6.3.34.1
0x0001	Length	See section 6.3.34.2
0x0002	Validity	See section 6.3.34.3
0x0003	Netlist MinSrev LSB	See section 6.3.34.4
0x0004	Netlist MinSrev MSB	See section 6.3.34.5

6.3.34.1 NetlistMinSrev Module Type - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x146	Valid values are: 0x146 = NetlistMinSrev Module Type

6.3.34.2 Length - 0x0001

Length of section

Bits	Field Name	Default NVM Value	Description
15:0	Block Length		Length in: 2 Bytes unit - 2 First Section -> Word: NetlistMinSrev -> NetlistMinSrev Module Type Last Section -> Word: NetlistMinSrev -> Netlist MinSrev MSB Section length in words Note: There are 21 MAC addresses to support switch mode.

6.3.34.3 Validity - 0x0002

Validity of MinSrev words

Bits	Field Name	Default NVM Value	Description
15:1	Reserved	0x0000	
0	Netlist	0x0	Netlist MinSrev valid.

6.3.34.4 Netlist MinSrev LSB - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	MinSrev LSB	0x0000	Netlist MinSrev

6.3.34.5 Netlist MinSrev MSB - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	MinSrev MSB	0x0000	Netlist MinSrev

6.3.35 OEM RoT Key Revocation Module Section Summary Table

Word Offset	Description	Reference
0x0000	Type	See section 6.3.35.1
0x0001	Length	See section 6.3.35.2

6.3.35.1 Type - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x14A	Valid values are: 0x14A = OEM RoT Key Revocation Module

6.3.35.2 Length - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit - 2 First Section -> Word: OEM RoT Key Revocation Module -> Type Last Section -> Word: OEM RoT Key Revocation Module -> OEM RoT Key Revocation Module CSS-signed

6.3.36 OEM RoT Key Revoke ISK Object CSS Header Section Summary Table

Word Offset	Description	Reference
0x0000	moduleTypeL	See section 6.3.36.1
0x0001	moduleTypeH	See section 6.3.36.2
0x0002	headerLenL	See section 6.3.36.3
0x0003	headerLenH	See section 6.3.36.4
0x0004	headerVersionL	See section 6.3.36.5
0x0005	headerVersionH	See section 6.3.36.6
0x0006	moduleIDL	See section 6.3.36.7
0x0007	moduleIDH	See section 6.3.36.8
0x0008	moduleVendorL	See section 6.3.36.9
0x0009	moduleVendorH	See section 6.3.36.10
0x000A	dateL	See section 6.3.36.11
0x000B	dateH	See section 6.3.36.12
0x000C	sizeL	See section 6.3.36.13
0x000D	sizeH	See section 6.3.36.14
0x000E	keySizeL	See section 6.3.36.15
0x000F	keySizeH	See section 6.3.36.16
0x0010	modulusSizeL	See section 6.3.36.17
0x0011	modulusSizeH	See section 6.3.36.18
0x0012	exponentSizeL	See section 6.3.36.19
0x0013	exponentSizeH	See section 6.3.36.20
0x0014	lad_srevL	See section 6.3.36.21
0x0015	lad_srevH	See section 6.3.36.22
0x0016	crypto_infoL	See section 6.3.36.23
0x0017	crypto_infoH	See section 6.3.36.24
0x0018	lad_fw_entry_offsetL	See section 6.3.36.25
0x0019	lad_fw_entry_offsetH	See section 6.3.36.26
0x001A	reserved2L	See section 6.3.36.27
0x001B	reserved2H	See section 6.3.36.28
0x001C	lad_image_unique_idL	See section 6.3.36.29
0x001D	lad_image_unique_idH	See section 6.3.36.30
0x001E	lad_module_idL	See section 6.3.36.31
0x001F	lad_module_idH	See section 6.3.36.32
0x0020 + 1*n, n=0...31	reserved	See section 6.3.36.33

6.3.36.1 moduleTypeL - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	moduleType	0x6	

6.3.36.2 moduleTypeH - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	moduleTypeH		

6.3.36.3 headerLenL - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	headerLenL	0x1A1	

6.3.36.4 headerLenH - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	headerLenH		

6.3.36.5 headerVersionL - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionL	0x00010000	

6.3.36.6 headerVersionH - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionH		

6.3.36.7 moduleIDL - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	moduleIDL	0x00	

6.3.36.8 moduleIDH - 0x0007

Bits	Field Name	Default NVM Value	Description
15	signMode	0x1	
14:0	moduleIDH		

6.3.36.9 moduleVendorL - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorL	0x00008086	

6.3.36.10 moduleVendorH - 0x0009

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorH		

6.3.36.11 dateL - 0x000A

0xMMDD

Bits	Field Name	Default NVM Value	Description
15:0	DateL	0x20130530	0xMMDD

6.3.36.12 dateH - 0x000B

0xYYYY

Bits	Field Name	Default NVM Value	Description
15:0	DateH		

6.3.36.13 sizeL - 0x000C

Bits	Field Name	Default NVM Value	Description
15:0	sizeL	0x1B6	

6.3.36.14 sizeH - 0x000D

Bits	Field Name	Default NVM Value	Description
15:0	sizeH		

6.3.36.15 keySizeL - 0x000E

Bits	Field Name	Default NVM Value	Description
15:0	keySizeL	0x60	

6.3.36.16 keySizeH - 0x000F

Bits	Field Name	Default NVM Value	Description
15:0	keySizeH		

6.3.36.17 modulusSizeL - 0x0010

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeL	0x60	

6.3.36.18 modulusSizeH - 0x0011

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeH		

6.3.36.19 exponentSizeL - 0x0012

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeL	0x1	

6.3.36.20 exponentSizeH - 0x0013

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeH		

6.3.36.21 lad_srevL - 0x0014

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevL	0x0	

6.3.36.22 lad_srevH - 0x0015

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevH		

6.3.36.23 crypto_infoL - 0x0016

Bits	Field Name	Default NVM Value	Description
15:8	hash algorithm	0x1	
7:0	sign mode	0x8	

6.3.36.24 crypto_infoH - 0x0017

Bits	Field Name	Default NVM Value	Description
15:8	reserved	0x0	
7:0	encryption algorithm	0x0	

6.3.36.25 lad_fw_entry_offsetL - 0x0018

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetL	0x34A	

6.3.36.26 lad_fw_entry_offsetH - 0x0019

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetH		

6.3.36.27 reserved2L - 0x001A

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.36.28 reserved2H - 0x001B

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.36.29 lad_image_unique_idL - 0x001C

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idL	0x0	

6.3.36.30 lad_image_unique_idH - 0x001D

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idH		

6.3.36.31 lad_module_idL - 0x001E

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idL	0xF0	Valid values are: 0x1 = EMP Image 0x2 = PE Image 0x3 = PCIe Analog 0x4 = PHY Analog 0x5 = Option ROM 0xB = Recovery 0xF0 = ISK Object

6.3.36.32 lad_module_idH - 0x001F

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idH		

6.3.36.33 reserved[n] (0x0020 + 1*n, n=0...31)

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.37 OEM RoT Key Revoke ISK Object Key and Signature Section Summary Table

Word Offset	Description	Reference
0x0000 + 1*n, n=0...191	RSA Public Key	See section 6.3.37.1
0x00C0 + 1*n, n=0...191	RSA Public Key Prime	See section 6.3.37.2
0x0180 + 1*n, n=0...191	RSA Public Key Second	See section 6.3.37.3
0x0240	RSA ExponentL	See section 6.3.37.4
0x0241	RSA ExponentH	See section 6.3.37.5
0x0242 + 1*n, n=0...191	Encrypted SHA384 Hash	See section 6.3.37.6

6.3.37.1 RSA Public Key[n] (0x0000 + 1*n, n=0...191)

Associated with the public portion of the Key Authentication Key used to validate this ISK.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.3.37.2 RSA Public Key Prime[n] (0x00C0 + 1*n, n=0...191)

Associated with the public portion of the Key Authentication Key used to validate this ISK.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key Prime		

6.3.37.3 RSA Public Key Second[n] (0x0180 + 1*n, n=0...191)

Associated with the public portion of the Key Authentication Key used to validate this ISK.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key Second		

6.3.37.4 RSA ExponentL - 0x0240

Associated with the public portion of the Key Authentication Key used to validate this ISK.

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentL	0x0	

6.3.37.5 RSA ExponentH - 0x0241

Associated with the public portion of the Key Authentication Key used to validate this ISK.

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentH	0x0	

6.3.37.6 Encrypted SHA384 Hash[n] (0x0242 + 1*n, n=0...191)

Bits	Field Name	Default NVM Value	Description
15:0	Encrypted SHA384 Hash	0x0	

6.3.38 OEM RoT Key Revoke ISK Object Auth Header Section Summary Table

Word Offset	Description	Reference
0x0000	Device Blank NVM Device ID	See section 6.3.38.1
0x0001	Max Module AreaL	See section 6.3.38.2
0x0002	Max Module AreaH	See section 6.3.38.3
0x0003	Current Module AreaL	See section 6.3.38.4
0x0004	Current Module AreaH	See section 6.3.38.5
0x0005	Reserved	See section 6.3.38.6
0x0006	Code Revision	See section 6.3.38.7
0x0007	Reserved Spare Word	See section 6.3.38.8

6.3.38.1 Device Blank NVM Device ID - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Device Blank NVM Device ID	0x57AC	

6.3.38.2 Max Module AreaL - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaL	0x36C	

6.3.38.3 Max Module AreaH - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaH	0x0	

6.3.38.4 Current Module AreaL - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaL	0x36C	

6.3.38.5 Current Module AreaH - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaH	0x0000	

6.3.38.6 Reserved - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	

6.3.38.7 Code Revision - 0x0006

Bits	Field Name	Default NVM Value	Description
15:8	Major Revision	0x0	
7:0	Minor Revision	0x0	

6.3.38.8 Reserved Spare Word - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	Reserved Spare Word	0x0	

6.3.39 OEM RoT Key Revoke ISK Object Section Summary Table

Word Offset	Description	Reference
0x0000	ISK ID	See section 6.3.39.1
0x0001 + 1*n, n=0...31	ISK SHA512 Hash	See section 6.3.39.2

6.3.39.1 ISK ID - 0x0000

Bits	Field Name	Default NVM Value	Description
15	Revocation Scheme	0x1	0x0 - Pseudo timer based revocation 0x1 - Explicit revocation
14:0	Key Revocation ID	0x0	Indicates the index of the fuse used to revoke this ISK.

6.3.39.2 ISK SHA512 Hash[n] (0x0001 + 1*n, n=0...31)

Bits	Field Name	Default NVM Value	Description
15:0	ISK SHA512 Hash	0x0	

6.3.40 OEM RoT Key Revocation Module CSS Header Section Summary Table

Word Offset	Description	Reference
0x0000	moduleTypeL	See section 6.3.40.1
0x0001	moduleTypeH	See section 6.3.40.2
0x0002	headerLenL	See section 6.3.40.3
0x0003	headerLenH	See section 6.3.40.4
0x0004	headerVersionL	See section 6.3.40.5
0x0005	headerVersionH	See section 6.3.40.6
0x0006	moduleIDL	See section 6.3.40.7
0x0007	moduleIDH	See section 6.3.40.8
0x0008	moduleVendorL	See section 6.3.40.9
0x0009	moduleVendorH	See section 6.3.40.10
0x000A	dateL	See section 6.3.40.11
0x000B	dateH	See section 6.3.40.12
0x000C	sizeL	See section 6.3.40.13
0x000D	sizeH	See section 6.3.40.14
0x000E	keySizeL	See section 6.3.40.15
0x000F	keySizeH	See section 6.3.40.16
0x0010	modulusSizeL	See section 6.3.40.17
0x0011	modulusSizeH	See section 6.3.40.18
0x0012	exponentSizeL	See section 6.3.40.19
0x0013	exponentSizeH	See section 6.3.40.20
0x0014	lad_srevL	See section 6.3.40.21
0x0015	lad_srevH	See section 6.3.40.22
0x0016	crypto_infoL	See section 6.3.40.23
0x0017	crypto_infoH	See section 6.3.40.24
0x0018	lad_fw_entry_offsetL	See section 6.3.40.25
0x0019	lad_fw_entry_offsetH	See section 6.3.40.26
0x001A	reserved2L	See section 6.3.40.27
0x001B	reserved2H	See section 6.3.40.28
0x001C	lad_image_unique_idL	See section 6.3.40.29
0x001D	lad_image_unique_idH	See section 6.3.40.30
0x001E	lad_module_idL	See section 6.3.40.31
0x001F	lad_module_idH	See section 6.3.40.32
0x0020 + 1*n, n=0...31	reserved	See section 6.3.40.33

6.3.40.1 moduleTypeL - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	moduleType	0x6	

6.3.40.2 moduleTypeH - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	moduleTypeH		

6.3.40.3 headerLenL - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	headerLenL	0x1A1	

6.3.40.4 headerLenH - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	headerLenH		

6.3.40.5 headerVersionL - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionL	0x00010000	

6.3.40.6 headerVersionH - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionH		

6.3.40.7 moduleIDL - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	moduleIDL	0x00	

6.3.40.8 moduleIDH - 0x0007

Bits	Field Name	Default NVM Value	Description
15	signMode	0x1	
14:0	moduleIDH		

6.3.40.9 moduleVendorL - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorL	0x00008086	

6.3.40.10 moduleVendorH - 0x0009

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorH		

6.3.40.11 dateL - 0x000A

0xMMDD

Bits	Field Name	Default NVM Value	Description
15:0	DateL	0x20130530	0xMMDD

6.3.40.12 dateH - 0x000B

0xYYYY

Bits	Field Name	Default NVM Value	Description
15:0	DateH		

6.3.40.13 sizeL - 0x000C

Bits	Field Name	Default NVM Value	Description
15:0	sizeL	0x1AF	

6.3.40.14 sizeH - 0x000D

Bits	Field Name	Default NVM Value	Description
15:0	sizeH		

6.3.40.15 keySizeL - 0x000E

Bits	Field Name	Default NVM Value	Description
15:0	keySizeL	0x60	

6.3.40.16 keySizeH - 0x000F

Bits	Field Name	Default NVM Value	Description
15:0	keySizeH		

6.3.40.17 modulusSizeL - 0x0010

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeL	0x60	

6.3.40.18 modulusSizeH - 0x0011

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeH		

6.3.40.19 exponentSizeL - 0x0012

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeL	0x1	

6.3.40.20 exponentSizeH - 0x0013

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeH		

6.3.40.21 lad_srevL - 0x0014

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevL	0x0	

6.3.40.22 lad_srevH - 0x0015

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevH		

6.3.40.23 crypto_infoL - 0x0016

Bits	Field Name	Default NVM Value	Description
15:8	hash algorithm	0x1	
7:0	sign mode	0x8	

6.3.40.24 crypto_infoH - 0x0017

Bits	Field Name	Default NVM Value	Description
15:12	reserved	0x0	
11:8	srev_flags	0x0	
7:0	encryption algorithm	0x0	

6.3.40.25 lad_fw_entry_offsetL - 0x0018

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetL	0x34A	

6.3.40.26 lad_fw_entry_offsetH - 0x0019

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetH		

6.3.40.27 reserved2L - 0x001A

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.40.28 reserved2H - 0x001B

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.40.29 lad_image_unique_idL - 0x001C

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idL	0x0	

6.3.40.30 lad_image_unique_idH - 0x001D

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idH		

6.3.40.31 lad_module_idL - 0x001E

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idL	0xF2	Valid values are: 0x1 = EMP Image 0x2 = PE Image 0x3 = PCIe Analog 0x4 = PHY Analog 0x5 = Option ROM 0xB = Recovery 0xF0 = ISK Object 0xF2 = Key Revocation Module

6.3.40.32 lad_module_idH - 0x001F

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idH		

6.3.40.33 reserved[n] (0x0020 + 1*n, n=0...31)

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.41 OEM RoT Key Revocation Module Key and Signature Section Summary Table

Word Offset	Description	Reference
0x0000 + 1*n, n=0...191	RSA Public Key	See section 6.3.41.1
0x00C0 + 1*n, n=0...191	RSA Public Key Prime	See section 6.3.41.2
0x0180 + 1*n, n=0...191	RSA Public Key Second	See section 6.3.41.3
0x0240	RSA ExponentL	See section 6.3.41.4
0x0241	RSA ExponentH	See section 6.3.41.5
0x0242 + 1*n, n=0...191	Encrypted SHA384 Hash	See section 6.3.41.6

6.3.41.1 RSA Public Key[n] (0x0000 + 1*n, n=0...191)

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.3.41.2 RSA Public Key Prime[n] (0x00C0 + 1*n, n=0...191)

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key Prime	0x0	

6.3.41.3 RSA Public Key Second[n] (0x0180 + 1*n, n=0...191)

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key Second	0x0	

6.3.41.4 RSA ExponentL - 0x0240

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentL	0x0	

6.3.41.5 RSA ExponentH - 0x0241

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentH	0x0	

6.3.41.6 Encrypted SHA384 Hash[n] (0x0242 + 1*n, n=0...191)

Bits	Field Name	Default NVM Value	Description
15:0	Encrypted SHA384 Hash	0x0	

6.3.42 OEM RoT Key Revocation Module Auth Header Section Summary Table

Word Offset	Description	Reference
0x0000	Device Blank NVM Device ID	See section 6.3.42.1
0x0001	Max Module AreaL	See section 6.3.42.2
0x0002	Max Module AreaH	See section 6.3.42.3
0x0003	Current Module AreaL	See section 6.3.42.4
0x0004	Current Module AreaH	See section 6.3.42.5
0x0005	Reserved	See section 6.3.42.6
0x0006	Code Revision	See section 6.3.42.7
0x0007	Reserved Spare Word	See section 6.3.42.8

6.3.42.1 Device Blank NVM Device ID - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Device Blank NVM Device ID	0x57AC	

6.3.42.2 Max Module AreaL - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaL	0x35E	

6.3.42.3 Max Module AreaH - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaH	0x0	

6.3.42.4 Current Module AreaL - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaL	0x35E	

6.3.42.5 Current Module AreaH - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaH	0x0000	

6.3.42.6 Reserved - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	

6.3.42.7 Code Revision - 0x0006

Bits	Field Name	Default NVM Value	Description
15:8	Major Revision	0x0	
7:0	Minor Revision	0x0	

6.3.42.8 Reserved Spare Word - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	Reserved Spare Word	0x0	

6.3.43 OEM RoT Key Revocation Module Data Section Summary Table

Word Offset	Description	Reference
0x0000	Revoke KAK	See section 6.3.43.1
0x0001	Reserved	See section 6.3.43.2
0x0002	Reserved	See section 6.3.43.3
0x0003 + 1*n, n=0...15	Revoke ISK	See section 6.3.43.4

6.3.43.1 Revoke KAK - 0x0000

Bits	Field Name	Default NVM Value	Description
15:3	RESERVED	0x0	
2	Revoke KAK2	0x0	If set, image allows for revocation of KAK2.
1	Revoke KAK1	0x0	If set, image allows for the revocation of KAK1.
0	Revoke KAK0	0x0	If set, image allows for the revocation of KAK0.

6.3.43.2 Reserved - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	RESERVED	0x0	

6.3.43.3 Reserved - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	RESERVED	0x0	

6.3.43.4 Revoke ISK[n] (0x0003 + 1*n, n=0...15)

Bits	Field Name	Default NVM Value	Description
15:0	Revoke ISK	0x0	If set, image allows for revocation of corresponding ISK.

6.3.44 Padding module Section Summary Table

Preserved Fields Area Header

Word Offset	Description	Reference
0x0000	Sub Module Type padding	See section 6.3.44.1
0x0001	Length	See section 6.3.44.2
0x0002	Padding	See section 6.3.44.3

6.3.44.1 Sub Module Type padding - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0xffff	Valid values are: 0xffff = Padding Module

6.3.44.2 Length - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit - 2 First Section -> Word: Padding module -> Sub Module Type padding Last Section -> Word: Padding module -> Padding

6.3.44.3 Padding - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	[New Field]	0xFFFF	

6.3.45 PCIR Type 1/2 Section Summary Table

Word Offset	Description	Reference
0x0000	Address Low at PCI_PWRDATA	See section 6.3.45.1
0x0001	Address High at PCI_PWRDATA	See section 6.3.45.2
0x0002	PCI_PWRDATA L	See section 6.3.45.3
0x0003	PCI_PWRDATA H	See section 6.3.45.4
0x0004	Address Low at PCI_REVID	See section 6.3.45.5
0x0005	Address High at PCI_REVID	See section 6.3.45.6
0x0006	PCI_REVID L	See section 6.3.45.7
0x0007	PCI_REVID H	See section 6.3.45.8
0x0008	Address Low at PCI_PMSUP	See section 6.3.45.9
0x0009	Address High at PCI_PMSUP	See section 6.3.45.10
0x000A	PCI_PMSUP L	See section 6.3.45.11
0x000B	PCI_PMSUP H	See section 6.3.45.12
0x000C	Address Low at PCI_PCIERR	See section 6.3.45.13
0x000D	Address High at PCI_PCIERR	See section 6.3.45.14
0x000E	PCI_PCIERR L	See section 6.3.45.15
0x000F	PCI_PCIERR H	See section 6.3.45.16

6.3.45.1 Address Low at PCI_PWRDATA - 0x0000

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_PWRDATA	0x11060	
3:0	Type	0x1	

6.3.45.2 Address High at PCI_PWRDATA - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_PWRDATA		

6.3.45.3 PCI_PWRDATA L - 0x0002

PCIe Power Data Register

Bits	Field Name	Default NVM Value	Description
15:0	PCI_PWRDATA L	0x0	

6.3.45.4 PCI_PWRDATA H - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	PCI_PWRDATA H		

6.3.45.5 Address Low at PCI_REVID - 0x0004

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_REVID	0x11098	
3:0	Type	0x1	

6.3.45.6 Address High at PCI_REVID - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_REVID		

6.3.45.7 PCI_REVID L - 0x0006

Revision ID

Bits	Field Name	Default NVM Value	Description
15:0	PCI_REVID L	0x0	

6.3.45.8 PCI_REVID H - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	PCI_REVID H		

6.3.45.9 Address Low at PCI_PMSUP - 0x0008

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_PMSUP	0x110A0	
3:0	Type	0x1	

6.3.45.10 Address High at PCI_PMSUP - 0x0009

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_PMSUP		

6.3.45.11 PCI_PMSUP L - 0x000A

PCIe Power Management Support

Bits	Field Name	Default NVM Value	Description
15:0	PCI_PMSUP L	0x00007397	

6.3.45.12 PCI_PMSUP H - 0x000B

Bits	Field Name	Default NVM Value	Description
15:0	PCI_PMSUP H		

6.3.45.13 Address Low at PCI_PCIERR - 0x000C

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_PCIERR	0x11140	
3:0	Type	0x1	

6.3.45.14 Address High at PCI_PCIERR - 0x000D

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_PCIERR		

6.3.45.15 PCI_PCIERR L - 0x000E

Error Enables

Bits	Field Name	Default NVM Value	Description
15:0	PCI_PCIERR L	0xE4FC	

6.3.45.16 PCI_PCIERR H - 0x000F

Bits	Field Name	Default NVM Value	Description
15:0	PCI_PCIERR H	0x000B	

6.3.46 Sideband Bus Autoload Section Summary Table

Word Offset	Description	Reference
0x0000	Module Length	See section 6.3.46.1
0x0001	IOW - Type F word 0	See section 6.3.46.2
0x0002	IOW - Type F word 1	See section 6.3.46.3
0x0003	IOW- Type F word 2	See section 6.3.46.4
0x0004	Manual Additions	See section 6.3.46.5

6.3.46.1 Module Length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 1 First Section -> Word: Sideband Bus Autoload -> Module Length Last Section -> Word: Sideband Bus Autoload -> Manual Additions

6.3.46.2 IOW - Type F word 0 - 0x0001

Bits	Field Name	Default NVM Value	Description
15:4	Subtype	0x0	
3:0	Type	0xF	

6.3.46.3 IOW - Type F word 1 - 0x0002

Bits	Field Name	Default NVM Value	Description
15	R	0x0	
14:12	TAG	0x0	
11	AL	0x0	
10:8	BAR	0x0	
7:4	Select ID	0x7	
3:0	Opcode	0x7	

6.3.46.4 IOW- Type F word 2 - 0x0003

Bits	Field Name	Default NVM Value	Description
15:12	Second Byte En	0x0	
11:8	First Byte En	0x2	
7:0	Function ID	0x0	

6.3.46.5 Manual Additions - 0x0004

Raw data module length: variable

6.3.47 EMP SR Settings Module Header Section Summary Table

This section contains the modes of operation of the EMP which shall be stored in the shadow RAM.

Word Offset	Description	Reference
0x0000	Section Header	See section 6.3.47.1
0x0001	SMBus Target Addresses	See section 6.3.47.2
0x0002 - 0x0004	Reserved	See section 6.3.47.3
0x0005	OEM capabilities	See section 6.3.47.4
0x0006	OEM technologies enabled	See section 6.3.47.5
0x0007	SR PF allocations pointer	See section 6.3.47.6
0x0008	Max PF and VF per Port	See section 6.3.47.7
0x0009	PF LAN Device ID	See section 6.3.47.8
0x000A - 0x000C	Reserved	See section 6.3.47.9
0x000D	VF LAN Device ID	See section 6.3.47.10
0x000E - 0x0010	Reserved	See section 6.3.47.11
0x0011	PF LAN Subsystem ID	See section 6.3.47.12
0x0012 - 0x0014	Reserved	See section 6.3.47.13
0x0015	VF LAN Subsystem ID	See section 6.3.47.14
0x0016 - 0x0018	Reserved	See section 6.3.47.15
0x0019 + 1*n, n=0...1	PFGEN_STATE	See section 6.3.47.16
0x001B	Reserved (OEM Current Settings Pointer)	See section 6.3.47.17
0x001C	Features enable	See section 6.3.47.18
0x001D	LLDP configuration pointer	See section 6.3.47.19
0x001E	Allowed SB targets	See section 6.3.47.20
0x001F	Allow 64 bits transactions	See section 6.3.47.21
0x0020	Allowed Opcodes	See section 6.3.47.22
0x0021	RX Hang Workaround Control 0	See section 6.3.47.23
0x0022	RX Hang Workaround Control 1	See section 6.3.47.24

6.3.47.1 Section Header - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Block Length		Length in: 2 Bytes unit - 1 First Section -> Word: EMP SR Settings Module Header -> Section Header Last Section -> Word: EMP SR Settings Module Header -> RX Hang Workaround Control 1 Section length in words

6.3.47.2 SMBus Target Addresses - 0x0001

Bits	Field Name	Default NVM Value	Description
15:9	SMBus 1 target Address	0x4A	Dual address mode only.
8	Reserved	0x0	
7:1	SMBus 0 target Address	0x49	
0	Reserved	0x0	

6.3.47.3 Reserved (0x0002 - 0x0004)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	Reserved.

6.3.47.4 OEM capabilities - 0x0005

Defines the OEM technologies supported in this device.

Bits	Field Name	Default NVM Value	Description
15	Skip Jack Island Support	0x0	0x0 -- Skip Jack Island support not enabled 0x1 -- Skip Jack Island support enabled Valid values are: 0x0 = Disabled - Skip Jack Island support not enabled 0x1 = Enabled - Skip Jack Island support enabled
14:8	Reserved	0x0	Reserved
7	Allow HII Port Disable	0x0	Determines whether the present HII configuration feature is present in the configuration menu. If this bit is set (1), the HII configuration feature shall be available; if the bit is clear (0), the HII configuration shall not be available. Valid values are: 0x0 = Disabled 0x1 = Enabled
6	OCBB over MCTP Capable	0x0	HP OCBB over MCTP Capable enable Valid values are: 0x0 = Disabled 0x1 = Capable
5	HP PLDM support Capable	0x0	HP PLDM support Capable Valid values are: 0x0 = Disabled 0x1 = Capable
4:0	Reserved	0x0	Reserved

6.3.47.5 OEM technologies enabled - 0x0006

Defines the OEM technologies enabled in this device.

Bits	Field Name	Default NVM Value	Description
15:14	Reserved	0x0	Reserved
13	Anti Rollback feature enable	0x0	If set, Anti Rollback feature enable Valid values are: 0x0 = Disabled 0x1 = Enabled
12	OCBB over MCTP	0x0	HP OCBB enable Valid values are: 0x0 = Disabled 0x1 = Enabled
11	HP PLDM support	0x0	Enable PLDM support Valid values are: 0x0 = Disabled 0x1 = Enabled
10:0	Reserved	0x0	Reserved

6.3.47.6 SR PF allocations pointer - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	PF allocations pointer	0xFFFF	

6.3.47.7 Max PF and VF per Port - 0x0008

Bits	Field Name	Default NVM Value	Description
15:8	Max VF per PF	0x80	Max number of VFs allocated to a PF.
7:3	Reserved	0x00	
2:0	Max PF per Port	0x2	Valid values are: 0x0 = 1 PF per Port 0x1 = 2 PFs per Port 0x2 = 4 PFs per Port 0x3 = 8 PFs per Port 0x4 = 16 PFs per Port

6.3.47.8 PF LAN Device ID - 0x0009

Bits	Field Name	Default NVM Value	Description
15:0	PF LAN Device ID	0x57AC	

6.3.47.9 Reserved (0x000A - 0x000C)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	

6.3.47.10 VF LAN Device ID - 0x000D

Bits	Field Name	Default NVM Value	Description
15:0	VF LAN Device ID	0x57AD	

6.3.47.11 Reserved (0x000E - 0x0010)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	

6.3.47.12 PF LAN Subsystem ID - 0x0011

Bits	Field Name	Default NVM Value	Description
15:0	PF LAN Subsystem ID	0x0	

6.3.47.13 Reserved (0x0012 - 0x0014)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	

6.3.47.14 VF LAN Subsystem ID - 0x0015

Bits	Field Name	Default NVM Value	Description
15:0	VF LAN Subsystem ID	0x0	

6.3.47.15 Reserved (0x0016 - 0x0018)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	

6.3.47.16 PFGEN_STATE[n] (0x0019 + 1*n, n=0...1)

Bits	Field Name	Default NVM Value	Description
15:12	Reserved	0x0	
11	PFGEN_STATE[1 + 2*n].PFSCEN	0x0	
10	PFGEN_STATE[1 + 2*n].PFLINKEN	0x1	
9	PFGEN_STATE[1 + 2*n].PFFCEN	0x1	
8	PFGEN_STATE[1 + 2*n].PFPEEN	0x0	
7:4	Reserved	0x0	
3	PFGEN_STATE[0 + 2*n].PFSCEN	0x0	
2	PFGEN_STATE[0 + 2*n].PFLINKEN	0x1	
1	PFGEN_STATE[0 + 2*n].PFFCEN	0x1	
0	PFGEN_STATE[0 + 2*n].PFPEEN	0x0	

6.3.47.17 Reserved (OEM Current Settings Pointer) - 0x001B

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	

6.3.47.18 Features enable - 0x001C

Describes support for various features

Bits	Field Name	Default NVM Value	Description
15	PHY 2 Autoload Enable	0x1	PHY Autoload Enable
14	PHY 1 Autoload Enable	0x1	PHY Autoload Enable
13	PHY 0 Autoload Enable	0x1	PHY Autoload Enable
12	PXE Mode No-Drop Policy Supported	0x0	The value indicates if the PXE Mode No-Drop policy is supported. 0 - No support. 1 - PXE Mode No-Drop is supported. This mode can be enabled by the Configure No-Drop Policy AQ command. Note: This field is preserved by Intel NVM Update Tool.
11:9	Switching mode	0x0	Reserved Valid values are: 0x0 = EVB switching 802.1Qbg switching

Bits	Field Name	Default NVM Value	Description
8	HII Version Check For Seamless Update Enable	0x0	New bit to enable the feature It will only be set for Dell and unset for all OEMGEN images Valid values are: 0x0 = Disabled 0x1 = Enabled Note: This field is preserved by Intel NVM update tool.
7:3	Reserved	0x0	Reserved Note: This field is preserved by Intel NVM update tool.
2	Reserved	0x1	Reserved Note: This field is preserved by Intel NVM update tool.
1	Reserved	0x0	Reserved Note: This field is preserved by Intel NVM update tool.
0	Non-BTS Switch mode	0x0	non-BTS mode Note: This field is preserved by Intel NVM update tool.

6.3.47.19 LLDP configuration pointer - 0x001D

Bits	Field Name	Default NVM Value	Description
15:0	LLDP configuration pointer	0xFFFF	

6.3.47.20 Allowed SB targets - 0x001E

The list of allowed sideband targets

Bits	Field Name	Default NVM Value	Description
15:12	Reserved	0x0	
11	SPI	0x0	
10	FC	0x0	
9	DSA	0x0	
8	LCB	0x1	
7	PCIW	0x1	
6	CAR	0x1	
5	IO Widget	0x1	
4:3	Reserved	0x0	
2	ETHW	0x1	
1:0	Reserved	0x0	

6.3.47.21 Allow 64 bits transactions - 0x001F

block/allow 64 bits transactions

Bits	Field Name	Default NVM Value	Description
15:1	Reserved		
0	Allow 64 bits transactions	0x0	

6.3.47.22 Allowed Opcodes - 0x0020

The list of allowed sideband opcodes

Bits	Field Name	Default NVM Value	Description
15	Reserved	0x0	
14	IOWr - IO Register Write (Non Posted)	0x0	
13	IOWr - IO Register Write (Posted)	0x0	
12	IORd - IO Register Read (Non Posted)	0x0	
11	CfgWr - Config Register Write (Non Posted)	0x1	
10	CfgWr - Config Register Write (Posted)	0x0	
9	CfgRd - Config Register Read (Non Posted)	0x1	
8	CRRd - Read Private Control Register (Non Posted)	0x0	
7	CRWr - Write Private Control Register (Non Posted)	0x0	
6	CRWr - Write Private Control Register (Posted)	0x0	
5	MBWr - Multiple Block Write (Non Posted)	0x0	
4	MBWr - Multiple Block Write (Posted)	0x0	
3	MBRd - Multiple Block Read (Non Posted)	0x0	
2	MWr - Register Write (Non Posted)	0x0	
1	MWr - Register Write (Posted)	0x1	
0	Mrd - Register Read (Non Posted)	0x1	

6.3.47.23 RX Hang Workaround Control 0 - 0x0021

Bits	Field Name	Default NVM Value	Description
15:14	reserved	0x0	
13:8	PE_ACK_REQ_PM_TH	0x3	
7:5	reserved	0x0	
4:0	REQ_WB_PM_TH	0x2	

6.3.47.24 RX Hang Workaround Control 1 - 0x0022

Bits	Field Name	Default NVM Value	Description
15	RX hang W/A Disable	0x0	
14:6	reserved	0x0	
5:0	RLAN_ACK_REQ_PM_TH	0x3	

6.3.48 GFID Module Section Summary Table

Preserved Fields Area Header

Word Offset	Description	Reference
0x0000	Length	See section 6.3.48.1
0x0001	GFID	See section 6.3.48.2
0x0002	GFID0	See section 6.3.48.3
0x0003	GFID1	See section 6.3.48.4
0x0004	GFID2	See section 6.3.48.5
0x0005	GFID3	See section 6.3.48.6
0x0006	GFID4	See section 6.3.48.7
0x0007	GFID5	See section 6.3.48.8
0x0008	GFID6	See section 6.3.48.9
0x0009	GFID7	See section 6.3.48.10
0x000A	GFID8	See section 6.3.48.11
0x000B	GFID9	See section 6.3.48.12
0x000C	GFID10	See section 6.3.48.13
0x000D	GFID11	See section 6.3.48.14
0x000E	GFID12	See section 6.3.48.15
0x000F	GFID13	See section 6.3.48.16
0x0010	GFID14	See section 6.3.48.17
0x0011	GFID15	See section 6.3.48.18
0x0012	GFID16	See section 6.3.48.19
0x0013	Original GFID	See section 6.3.48.20
0x0014	Original GFID0	See section 6.3.48.21
0x0015	Original GFID1	See section 6.3.48.22
0x0016	Original GFID2	See section 6.3.48.23
0x0017	Original GFID3	See section 6.3.48.24
0x0018	Original GFID4	See section 6.3.48.25
0x0019	Original GFID5	See section 6.3.48.26
0x001A	Original GFID6	See section 6.3.48.27
0x001B	Original GFID7	See section 6.3.48.28
0x001C	Original GFID8	See section 6.3.48.29
0x001D	Original GFID9	See section 6.3.48.30
0x001E	Original GFID10	See section 6.3.48.31
0x001F	Original GFID11	See section 6.3.48.32
0x0020	Original GFID12	See section 6.3.48.33
0x0021	Original GFID13	See section 6.3.48.34
0x0022	Original GFID14	See section 6.3.48.35

Word Offset	Description	Reference
0x0023	Original GFID15	See section 6.3.48.36
0x0024	Original GFID16	See section 6.3.48.37

6.3.48.1 Length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit - 1 First Section -> Word: GFID Module -> Length Last Section -> Word: GFID Module -> Original GFID16

6.3.48.2 GFID - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	GFID	0x157	

6.3.48.3 GFID0 - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	GFID	0x57AC	

6.3.48.4 GFID1 - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	GFID	0x0	

6.3.48.5 GFID2 - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	GFID	0x0	

6.3.48.6 GFID3 - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	GFID	0x0	

6.3.48.7 GFID4 - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	GFID	0x0	

6.3.48.8 GFID5 - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	GFID	0x0	

6.3.48.9 GFID6 - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	GFID	0x0	

6.3.48.10 GFID7 - 0x0009

Bits	Field Name	Default NVM Value	Description
15:0	GFID	0x0	

6.3.48.11 GFID8 - 0x000A

Bits	Field Name	Default NVM Value	Description
15:0	GFID	0x0	

6.3.48.12 GFID9 - 0x000B

Bits	Field Name	Default NVM Value	Description
15:0	GFID	0x0	

6.3.48.13 GFID10 - 0x000C

Bits	Field Name	Default NVM Value	Description
15:0	GFID	0x0	

6.3.48.14 GFID11 - 0x000D

Bits	Field Name	Default NVM Value	Description
15:0	GFID	0x0	

6.3.48.15 GFID12 - 0x000E

Bits	Field Name	Default NVM Value	Description
15:0	GFID	0x0	

6.3.48.16 GFID13 - 0x000F

Bits	Field Name	Default NVM Value	Description
15:0	GFID	0x0	

6.3.48.17 GFID14 - 0x0010

Bits	Field Name	Default NVM Value	Description
15:0	GFID	0x0	

6.3.48.18 GFID15 - 0x0011

Bits	Field Name	Default NVM Value	Description
15:0	GFID	0x0	

6.3.48.19 GFID16 - 0x0012

Bits	Field Name	Default NVM Value	Description
15:0	GFID	0x0	

6.3.48.20 Original GFID - 0x0013

Bits	Field Name	Default NVM Value	Description
15:0	Original GFID	0x57AC	

6.3.48.21 Original GFID0 - 0x0014

Bits	Field Name	Default NVM Value	Description
15:0	Original GFID	0x57AC	

6.3.48.22 Original GFID1 - 0x0015

Bits	Field Name	Default NVM Value	Description
15:0	Original GFID	0x0	

6.3.48.23 Original GFID2 - 0x0016

Bits	Field Name	Default NVM Value	Description
15:0	Original GFID	0x0	

6.3.48.24 Original GFID3 - 0x0017

Bits	Field Name	Default NVM Value	Description
15:0	Original GFID	0x0	

6.3.48.25 Original GFID4 - 0x0018

Bits	Field Name	Default NVM Value	Description
15:0	Original GFID	0x0	

6.3.48.26 Original GFID5 - 0x0019

Bits	Field Name	Default NVM Value	Description
15:0	Original GFID	0x0	

6.3.48.27 Original GFID6 - 0x001A

Bits	Field Name	Default NVM Value	Description
15:0	Original GFID	0x0	

6.3.48.28 Original GFID7 - 0x001B

Bits	Field Name	Default NVM Value	Description
15:0	Original GFID	0x0	

6.3.48.29 Original GFID8 - 0x001C

Bits	Field Name	Default NVM Value	Description
15:0	Original GFID	0x0	

6.3.48.30 Original GFID9 - 0x001D

Bits	Field Name	Default NVM Value	Description
15:0	Original GFID	0x0	

6.3.48.31 Original GFID10 - 0x001E

Bits	Field Name	Default NVM Value	Description
15:0	Original GFID	0x0	

6.3.48.32 Original GFID11 - 0x001F

Bits	Field Name	Default NVM Value	Description
15:0	Original GFID	0x0	

6.3.48.33 Original GFID12 - 0x0020

Bits	Field Name	Default NVM Value	Description
15:0	Original GFID	0x0	

6.3.48.34 Original GFID13 - 0x0021

Bits	Field Name	Default NVM Value	Description
15:0	Original GFID	0x0	

6.3.48.35 Original GFID14 - 0x0022

Bits	Field Name	Default NVM Value	Description
15:0	Original GFID	0x0	

6.3.48.36 Original GFID15 - 0x0023

Bits	Field Name	Default NVM Value	Description
15:0	Original GFID	0x0	

6.3.48.37 Original GFID16 - 0x0024

Bits	Field Name	Default NVM Value	Description
15:0	Original GFID	0x0	

6.3.49 Manageability Module Header Section Summary Table

This section contains parameters related to the manageability functionality such as connection type and others, It also points to subsections configuring the filters and the sideband interfaces.

Word Offset	Description	Reference
0x0000	Section Length	See section 6.3.49.1
0x0001	Common Manageability Parameters	See section 6.3.49.2
0x0002	Common Manageability Parameters 2	See section 6.3.49.3
0x0003	PLDM Control Word	See section 6.3.49.4
0x0004	RDE Control word	See section 6.3.49.5
0x0005	OCP NIC parameters obsolete	See section 6.3.49.6
0x0006	OCP NIC Parameters 2	See section 6.3.49.7
0x0007	Sideband Configuration Pointer	See section 6.3.49.8
0x0008	Reserved	See section 6.3.49.9
0x0009	Traffic Types Parameters	See section 6.3.49.10
0x000A	OEM Section Pointer	See section 6.3.49.11

6.3.49.1 Section Length - 0x0000

The length of the section in words. Note that section length does not include a count for the section length word.

Bits	Field Name	Default NVM Value	Description
15:0	Section Length		Length in: 2 Bytes unit - 1 First Section -> Word: Manageability Module Header -> Section Length Last Section -> Word: Manageability Module Header -> OEM Section Pointer

6.3.49.2 Common Manageability Parameters - 0x0001

Bits	Field Name	Default NVM Value	Description
15	Reserved	0x0	Reserved
14:13	Reserved	0x1	Reserved
12:11	Reserved	0x0	Reserved
10:8	Manageability Pass Through Mode	0x2	Valid values are: 0x0 = None 0x2 = Pass through (PT) mode
7:5	Reserved	0x0	
4	Force TCO Reset Disable	0x1	If cleared, allows the BMC to do a Global reset of the device using the Force TCO SMBus or NC-SI commands Valid values are: 0x0 = Enable Force TCO reset. 0x1 = Disable Force TCO reset.
3	Reserved	0x0	

Bits	Field Name	Default NVM Value	Description
2	OS2BMC Capable	0x1	0b = Disable, 1b = Enable. Valid values are: 0x0 = Disabled 0x1 = Enabled
1:0	Reserved	0x0	Reserved

6.3.49.3 Common Manageability Parameters 2 - 0x0002

Bits	Field Name	Default NVM Value	Description
15:12	Reserved	0x0	Reserved
11	Multi-Drop NC-SI	0x1	Valid values are: 0x0 = Point-to-point 0x1 = Multi-drop
10	Reserved		
9	EMP_LINK_ON	0x1	Copy of the AUX GSCRL.EN_PHY_D3_DR bit for EMP use Valid values are: 0x0 = Disable 0x1 = Enable
8	Redfish support for PLDM (type 6)	0x1	Valid values are: 0x0 = Disabled 0x1 = Enabled
7	FW update over PLDM support	0x0	Support for PLDM base is defined as bit 7 bit 6 (FW update or monitoring and control support) Valid values are: 0x0 = Disabled 0x1 = Enabled
6	PLDM monitoring and control	0x0	Support for PLDM base is defined as bit 7 bit 6 (FW update or monitoring and control support) Valid values are: 0x0 = Disabled 0x1 = Enabled
5	OEM over MCTP support	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
4	NC-SI over MCTP support	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
3:2	Reserved	0x0	Reserved
1	LAN1_FT_CO_ISOL_DISS	0x1	Allow isolation of port 1 per BMC request Valid values are: 0x0 = Enabled 0x1 = Disabled
0	LAN0_FT_CO_ISOL_DISS	0x1	Allow isolation of port 0 per BMC request Valid values are: 0x0 = Enabled 0x1 = Disabled

6.3.49.4 PLDM Control Word - 0x0003

Bits	Field Name	Default NVM Value	Description
15:14	PLDM Offset	0x0	
13:9	Reserved	0x0	
8	Expose NIC Composite State sensors	0x1	
7	Expose power sensors	0x1	
6	Expose thermal trip state sensors	0x1	
5	Expose Configuration and Configuration Change	0x1	
4	Expose Presence state	0x1	
3	Expose Health State	0x1	
2	Expose Thermal Sensors	0x1	
1	Expose Link Speed	0x1	
0	Expose Link Status	0x1	

6.3.49.5 RDE Control word - 0x0004

Bits	Field Name	Default NVM Value	Description
15:4	Reserved	0x0	Reserved for other schemas including OEM schema extensions.
3	EthernetInterface v1.4.0	0x1	Ethernet Interface schema.
2:1	Reserved	0x0	Reserved
0	ACD	0x1	ACD collection.

6.3.49.6 OCP NIC parameters obsolete - 0x0005

Bits	Field Name	Default NVM Value	Description
15:7	Reserved	0x0	
6	FAN_ON_AUX valid obsolete	0x0	
5	FAN_ON_AUX value obsolete	0x0	
4:0	FAN_ON_AUX SDP obsolete	0x0	valid only if FAN_ON_AUX valid = 1

6.3.49.7 OCP NIC Parameters 2 - 0x0006

Bits	Field Name	Default NVM Value	Description
15:10	Reserved	0x0	
9	OCP compliance indicator	0x0	0x0 = for standup card 0x1 = for OCP compliant card
8:4	Instance ID SDP	0x0	This field determines the SDP that is used for the Instance ID on a dual-device NIC. This field is only valid if the Instance ID from SDP field is 1b.
3	Instance ID from SDP	0x0	This bit determines whether the instance ID should be taken from an SDP. 0b - There is only one device on the NIC. 1b - There are two devices on the NIC. The instance ID of this device is taken from the SDP defined by the Instance ID SDP field.
2	FAN_ON_AUX present	0x0	
1	TEMP_CRIT_N present	0x0	
0	TEMP_WARN_N present	0x0	

6.3.49.8 Sideband Configuration Pointer - 0x0007

This module is long by 28B and must be mapped in the first valid 4KB sector of the Flash.

Bits	Field Name	Default NVM Value	Description
15:0	Sideband Configuration Pointer	0xFFFF	

Field Sideband Configuration Pointer points to Sideband Configuration Structure Section. For Sideband Configuration Structure inner structure See [section 6.3.50](#)

6.3.49.9 Reserved - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	

6.3.49.10 Traffic Types Parameters - 0x0009

Bits	Field Name	Default NVM Value	Description
15:4	Reserved	0x0	
3:2	Port 1 traffic types	0x1	Defines which type of traffic can flow to and from primary BMC connection on port 1. The traffic types defined by this field are enabled by the Manageability Mode field and the OS2BMC Capable bit in the Common Manageability Parameters 1 NVM word. Valid values are: 0x0 = Reserved 0x1 = Network to BMC traffic only 0x2 = OS2BMC traffic only 0x3 = Both Network to BMC traffic and OS2BMC traffic
1:0	Port 0 traffic types	0x1	Defines which type of traffic can flow to and from primary BMC connection on port 0. The traffic types defined by this field are enabled by the Manageability Mode field and the OS2BMC Capable bit in the Common Manageability Parameters 1 NVM word. Valid values are: 0x0 = Reserved 0x1 = Network to BMC traffic only 0x2 = OS2BMC traffic only 0x3 = Both Network to BMC traffic and OS2BMC traffic

6.3.49.11 OEM Section Pointer - 0x000A

Bits	Field Name	Default NVM Value	Description
15:0	OEM Section Pointer	0xFFFF	

6.3.50 Sideband Configuration Structure Section Summary Table

This section describes the setting of the different pass through interfaces (SMBus, NC-SI & MCTP).

Word Offset	Description	Reference
0x0000	Section Length	See section 6.3.50.1
0x0001	SMBus Maximum Fragment Size	See section 6.3.50.2
0x0002	SMBus Notification Timeout and Flags	See section 6.3.50.3
0x0003	NC-SI Configuration 1	See section 6.3.50.4
0x0004	NC-SI Configuration 2	See section 6.3.50.5
0x0005	NC-SI Flow Control XOFF	See section 6.3.50.6
0x0006	NC-SI Flow Control XON	See section 6.3.50.7
0x0007	NC-SI HW Arbitration Configuration	See section 6.3.50.8
0x0008 - 0x000C	MCTP UUID - Time Low LSB	See section 6.3.50.9
0x000D	OEM IANA	See section 6.3.50.10
0x000E	NC-SI over MCTP Message Types	See section 6.3.50.11
0x000F	NC-SI over MCTP configuration	See section 6.3.50.12
0x0010	MCTP Rate Limiter Config 1	See section 6.3.50.13
0x0011	MCTP Rate Limiter Config 2	See section 6.3.50.14

6.3.50.1 Section Length - 0x0000

The length of the section in words. Note that section length does not include a count for the section length word.

Bits	Field Name	Default NVM Value	Description
15:0	Section Length		Length in: 2 Bytes unit - 1 First Section -> Word: Sideband Configuration Structure -> Section Length Last Section -> Word: Sideband Configuration Structure -> MCTP Rate Limiter Config 2

6.3.50.2 SMBus Maximum Fragment Size - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Fragment size	0x0020	SMBus Maximum Fragment Size (bytes). Supported range is between 32 and 240 bytes. Note: In MCTP mode, this value should be set to 0x45 (64 bytes payload + 5 bytes of MCTP header)

6.3.50.3 SMBus Notification Timeout and Flags - 0x0002

Bits	Field Name	Default NVM Value	Description
15:8	SMBus Notification Timeout (ms)	0xFF	
7:6	SMBus Connection Speed	0x0	Valid values are: 0x0 = Standard SMBus connection 0x1 = 400 Kbps fast i2c 0x2 = 1 Mbps fast+ i2c 0x3 = Reserved
5	SMBus Block Read command	0x0	Valid values are: 0x0 = Block read command is 0xC0 0x1 = Block read command is 0xD0
4	Reserved	0x0	
3	Enable Fairness Arbitration	0x1	MCTP over SMBus feature. 0b = Disable fairness arbitration. 1b = Enable fairness arbitration. Valid values are: 0x0 = Disabled 0x1 = Enabled
2	Disable SMBus ARP Functionality	0x0	Valid values are: 0x0 = SMBus ARP Enabled 0x1 = SMBus ARP Disabled
1	SMBus ARP PEC	0x1	SMBus ARP PEC Should be set in MCTP modes. Valid values are: 0x0 = Disable PEC - Disable SMBus ARP PEC. 0x1 = Enable PEC - Enable SMBus ARP PEC.
0	SMBus Transaction PEC	0x0	SMBus Transactions PEC Should be set in MCTP modes. Valid values are: 0x0 = Disable PEC - If this bit is cleared, PEC is not added to master write or slave read transactions, a slave write transaction with PEC is dropped. 0x1 = Enable PEC - If this bit is set, PEC is added for master SMBus write transactions. a PEC is added to slave read transactions and can be received in slave write transaction.

6.3.50.4 NC-SI Configuration 1 - 0x0003

Bits	Field Name	Default NVM Value	Description
15	NC-SI Channel to Port Mapping Table Valid	0x0	Valid values are: 0x0 = Table Invalid Use default algorithm described in Channel ID mapping section 0x1 = Table Valid Use the mapping defined in this word.
14	Flow Control	0x0	Valid values are: 0x0 = NC-SI flow control disable 0x1 = NC-SI flow control enable
13:10	NC-SI Version	0x2	Supported NC-SI version Valid values are: 0x0 = NC-SI 1.0.1 0x1 = NC-SI 1.1 0x2 = NC-SI 1.2

Bits	Field Name	Default NVM Value	Description
9	NC-SI HW arbitration enable	0x0	Valid values are: 0x0 = Not supported 0x1 = Supported
8	Reserved	0x1	Reserved
7:5	Package ID	0x0	Meaningful only when bit 15 of NC-SI Configuration 2 word (offset 0x07) is cleared.
4:0	Reserved	0x0	Reserved - must be zero

6.3.50.5 NC-SI Configuration 2 - 0x0004

Bits	Field Name	Default NVM Value	Description
15:14	Read NC-SI Package ID from SDP	0x00	Read NC-SI Package ID from: Valid values are: 0x0 = PackageID from NVM Package ID is read from NVM Package ID field in the NC-SI Configuration 0x1 = PackageID from NVM and SDPs #0 and #1 The package ID is (NVM Package ID[2], SDP#1 value, SDP#0 value). 0x2 = PackageID from NVM and SDP#1 The package ID is (NVM Package ID[2], SDP#1 value, NVM Package ID[0]). 0x3 = Reserved
13:9	PackageID SDP 0	0x0	Defines the SDP#1 from which the NC-SI package ID[1] is taken
8:4	PackageID SDP 1	0x0	Defines the SDP#0 from which the NC-SI package ID[0] is taken
3:0	Max XOFF Renewal	0x03	NC-SI Flow Control MAX XOFF Renewal (# of XOFF renewals allowed) 0x0 = Disabled. Unlimited number of XOFF frames can be sent. 0x1 = Up to 2 consecutive XOFFs frames can be sent by the device. 0x2 = Up to 3 consecutive XOFFs frames can be sent by the device. ... 0xF = Up to 16 consecutive XOFFs frames can be sent by the device.

6.3.50.6 NC-SI Flow Control XOFF - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	XOFF Threshold	0x12C0	Tx buffer watermark for sending a XOFF NC-SI flow control packet in bytes. The XOFF Threshold value refers to the occupied space in the buffer. The value should be 16 bytes aligned. Notes: - Field relevant for NC-SI operation mode only. - To support a maximum packet size of 1.5 KB, the value programmed assuming a Tx buffer size of 8 KB value of field should be 0x12C0 (4,800 bytes).

6.3.50.7 NC-SI Flow Control XON - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	XON Threshold	0x1340	Tx buffer water mark for sending a XON NC-SI flow control packet in bytes. The XON Threshold value refers to the available space in the TX buffer. The value should be 16 bytes aligned. Notes: - Field relevant for NC-SI operation mode only. - To support maximum packet size of 1.5 KB, the value programmed should be a positive value that equals: Buffer size - XOFF Threshold (refer to Section 6.5.5.7) + 1536 bytes. Assuming a TX Buffer size is 8 KB and the XOFF Threshold is 4800 bytes value of field should be 0x1340 (4,928 bytes).

6.3.50.8 NC-SI HW Arbitration Configuration - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	TOKEN Timeout	0xA000	NC-SI HW-Arbitration TOKEN Timeout (in NC-SI REF_CLK cycles - 20 ns). Setting value to 0 disables the timeout mechanism.

6.3.50.9 Reserved (0x0008 - 0x000C)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	Reserved.

6.3.50.10 OEM IANA - 0x000D

Bits	Field Name	Default NVM Value	Description
15:0	OEM IANA	0x0	Identifies the OEM targeted by this image. This impacts exposure of OEM specific features: - Dell: If the value is 0x2A2, the Dell commands described in the Dell support appendix are supported. In this case, the CEM enable bit in OEM Technologies word should be set also. - CISCO: If the value is 0x9, then the System Firmware Update Control filed in Set Intel management control command (Intel command 0x20) is supported. - HPE: If the value is 0xB, then HPE NC-SI OEM commands are accepted and OCBB & DCi command are supported. In this case, the appropriate bits in OEM Capabilities and OEM Technologies words should be set also. For Dell and HPE, the device will accept additional OEM commands with this IANA number. These commands are accepted only if the OEM NVM Structure is present in the PFA. The set of commands accepted depends on the IANA value in this field.

6.3.50.11 NC-SI over MCTP Message Types - 0x000E

Bits	Field Name	Default NVM Value	Description
15:8	NC-SI Control Message type	0x2	Defines the MCTP message type used to identify NC-SI Control Packets
7:0	NC-SI Pass Through Message type	0x3	Defines the MCTP message type used to identify NC-SI pass through packets

6.3.50.12 NC-SI over MCTP configuration - 0x000F

Bits	Field Name	Default NVM Value	Description
15:7	Reserved	0x0	Reserved
6	Simplified MCTP	0x0	If set, only SOM & EOM bits are used for the reassembly process. Relevant only in SMBus mode
5	Disable ACLs	0x0	If set, the ACLs on the PCIe VDMs are disabled.
4:0	Reserved	0x0	Reserved

6.3.50.13 MCTP Rate Limiter Config 1 - 0x0010

MCTP rate limiter configuration for first channel

Bits	Field Name	Default NVM Value	Description
15:0	MCTP Rate	0x9C40	Defines the number of cycles between accesses of the MCTP send client to the memory arbiter. Current value assumes a clock of 312.5 MHz and a bus width of 128 bit. This value provides a bit rate of 1 Mbps.

6.3.50.14 MCTP Rate Limiter Config 2 - 0x0011

MCTP rate limiter configuration for first channel

Bits	Field Name	Default NVM Value	Description
15	Decision Point	0x0	Defines if, when credits are available, a full MCTP message is sent or a single VDM is sent. Valid values are: 0x0 = Per VDM 0x1 = Per Packet
14:0	MCTP max credits	0x5	Defines the maximum number of 16 bytes credit that can be accumulated. These credits include the VDM header line (one line for each 64 byte VDM).

6.3.51 OEM Section Section Summary Table

This section is the header of the OEM specific data identifying the OEM for which this data is defined.

Word Offset	Description	Reference
0x0000	Section Length	See section 6.3.51.1
0x0001	OEM header	See section 6.3.51.2
0x0002	Reserved	See section 6.3.51.3

6.3.51.1 Section Length - 0x0000

The length of the section in words. Note that section length does not include a count for the section length word.

Bits	Field Name	Default NVM Value	Description
15:0	Section Length		Length in: 2 Bytes unit - 2 First Section -> Word: OEM Section -> Section Length Last Section -> Word: OEM Section -> Reserved

6.3.51.2 OEM header - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	OEM identifier	0x02A2	Identify the OEM for which this section is defined. Should be equal to the OEM IANA value.

6.3.51.3 Reserved - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	

6.3.52 Auto Generated Pointers Module Section Summary Table

Pointers to Type 1/2 words used by EMP and SW

Word Offset	Description	Reference
0x0000	Module Length	See section 6.3.52.1
0x0001	Pointer to PCI_CAPSUP Section	See section 6.3.52.2
0x0002	Pointer to PCI_CAPSUP Offset	See section 6.3.52.3
0x0003	Pointer to PCI_SUBSYSID Section	See section 6.3.52.4
0x0004	Pointer to PCI_SUBSYSID Offset	See section 6.3.52.5
0x0005	Pointer to PCI_VENDORID Section	See section 6.3.52.6
0x0006	Pointer to PCI_VENDORID Offset	See section 6.3.52.7
0x0007	Pointer to PCI_REVID Section	See section 6.3.52.8
0x0008	Pointer to PCI_REVID Offset	See section 6.3.52.9
0x0009	Pointer to PCI_PFDEVID_PF0 Section	See section 6.3.52.10
0x000A	Pointer to PCI_PFDEVID_PF0 Offset	See section 6.3.52.11
0x000B	Pointer to PCI_PFDEVID_PF1 Section	See section 6.3.52.12
0x000C	Pointer to PCI_PFDEVID_PF1 Offset	See section 6.3.52.13
0x000D	Pointer to PCI_VFDEVID_PF0 Section	See section 6.3.52.14
0x000E	Pointer to PCI_VFDEVID_PF0 Offset	See section 6.3.52.15
0x000F	Pointer to PCI_VFDEVID_PF1 Section	See section 6.3.52.16
0x0010	Pointer to PCI_VFDEVID_PF1 Offset	See section 6.3.52.17

6.3.52.1 Module Length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 1 First Section -> Word: Auto Generated Pointers Module -> Module Length Last Section -> Word: Auto Generated Pointers Module -> Pointer to PCI_VFDEVID_PF1 Offset

6.3.52.2 Pointer to PCI_CAPSUP Section - 0x0001

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	Pointer to PCI_CAPSUP Section	0x0	

6.3.52.3 Pointer to PCI_CAPSUP Offset - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to PCI_CAPSUP Offset	0x16	

6.3.52.4 Pointer to PCI_SUBSYSID Section - 0x0003

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	Pointer to PCI_SUBSYSID Section		

6.3.52.5 Pointer to PCI_SUBSYSID Offset - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to PCI_SUBSYSID Offset	0x22	

6.3.52.6 Pointer to PCI_VENDORID Section - 0x0005

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	Pointer to PCI_VENDORID Section		

6.3.52.7 Pointer to PCI_VENDORID Offset - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to PCI_VENDORID Offset	0x36	

6.3.52.8 Pointer to PCI_REVID Section - 0x0007

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	Pointer to PCI_REVID Section	0x0	

Field Pointer to PCI_REVID Section points to PCIR Type 1/2 Section. For PCIR Type 1/2 inner structure See [section 6.3.45](#)

6.3.52.9 Pointer to PCI_REVID Offset - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to PCI_REVID Offset	0x06	

6.3.52.10 Pointer to PCI_PFDEVID_PFO Section - 0x0009

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	Pointer to PCI_PFDEVID Section		

6.3.52.11 Pointer to PCI_PFDEVID_PFO Offset - 0x000A

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to PCI_PFDEVID Offset	0x3A	

6.3.52.12 Pointer to PCI_PFDEVID_PF1 Section - 0x000B

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	Pointer to PCI_PFDEVID Section		

6.3.52.13 Pointer to PCI_PFDEVID_PF1 Offset - 0x000C

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to PCI_PFDEVID Offset	0x4C	

6.3.52.14 Pointer to PCI_VFDEVID_PF0 Section - 0x000D

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	Pointer to PCI_VFDEVID Section		

6.3.52.15 Pointer to PCI_VFDEVID_PF0 Offset - 0x000E

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to PCI_VFDEVID Offset	0x42	

6.3.52.16 Pointer to PCI_VFDEVID_PF1 Section - 0x000F

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	Pointer to PCI_VFDEVID Section		

6.3.52.17 Pointer to PCI_VFDEVID_PF1 Offset - 0x0010

Bits	Field Name	Default NVM Value	Description
15:0	Pointer to PCI_VFDEVID Offset	0x54	

6.3.53 MAC Reset 0 Registers Autoload Module Section Summary Table

Default setup to registers and internal memories that load on MAC reset events

Word Offset	Description	Reference
0x0000	Module Length	See section 6.3.53.1
0x0001	Starting Address Low at MACC	See section 6.3.53.2
0x0002	Starting Address High at MACC	See section 6.3.53.3
0x0003	Data low of MACC	See section 6.3.53.4
0x0004	Data high of MACC	See section 6.3.53.5
0x0005	Starting Address Low at MFLCN	See section 6.3.53.6
0x0006	Starting Address High at MFLCN	See section 6.3.53.7
0x0007	Data low of MFLCN	See section 6.3.53.8
0x0008	Data high of MFLCN	See section 6.3.53.9

6.3.53.1 Module Length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 1 First Section -> Word: MAC Reset 0 Registers Autoload Module -> Module Length Last Section -> Word: MAC Reset 0 Registers Autoload Module -> Data high of MFLCN

6.3.53.2 Starting Address Low at MACC - 0x0001

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of MACC	0x4330	
3:0	Type	0x1	

6.3.53.3 Starting Address High at MACC - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of MACC		

6.3.53.4 Data low of MACC - 0x0003

Bits	Field Name	Default NVM Value	Description
15:12	FIFOLT	0x3	FIFO Low Threshold. Determines the low threshold of the MAC elastic FIFO. Mapped to MACC.FIFOHT.
11:8	FIFOHT	0xD	FIFO High Threshold. Determines the high threshold of the MAC elastic FIFO. Mapped to MACC.FIFOLT.
7	Swap Rx Control	0x0	Swap the MAC internal Rx XGMII controls. 0b = Swap disabled, normal mode. 1b = Swap enabled. Mapped to MACC.Swap Rx Control. Valid values are: 0x0 = Swap disabled, normal mode. 0x1 = Swap enabled
6	Swap Rx Data	0x0	Swap the MAC internal Rx XGMII lanes. 0b = Swap disabled, normal mode. 1b = Swap enabled. Mapped to MACC.Swap Rx Data Valid values are: 0x0 = Swap disabled, normal mode. 0x1 = Swap enabled
5	Swizzle Rx Data	0x0	Swizzle the bytes in all 4 MAC internal Rx XGMII lanes. 0b = Swizzle disabled, normal mode. 1b = Swizzle enabled. Mapped to MACC.Swizzle Rx Data Valid values are: 0x0 = Swizzle disabled, normal mode. 0x1 = Swizzle enabled
4	Swap Tx Control	0x0	Swap the MAC internal Tx XGMII controls. 0b = Swap disabled, normal mode. 1b = Swap enabled. Mapped to MACC.Swap Tx Control Valid values are: 0x0 = Swap disabled, normal mode. 0x1 = Swap enabled
3	Swap Tx Data	0x0	Swap the MAC internal Tx XGMII lanes. 0b = Swap disabled, normal mode. 1b = Swap enabled. Mapped to MACC.Swap Tx Data Valid values are: 0x0 = Swap disabled, normal mode. 0x1 = Swap enabled
2	Swizzle Tx Data	0x0	Swizzle the bytes in all 4 MAC internal Tx XGMII lanes. 0b = Swizzle disabled, normal mode. 1b = Swizzle enabled. Mapped to MACC.Swizzle Tx Data Valid values are: 0x0 = Swizzle disabled, normal mode. 0x1 = Swizzle enabled
1	MAC_RX2TX_LPBK_EN	0x0	Enable MAC Rx to Tx loopback. 0 - No loopback, normal mode. 1 - Loopback enabled. Transmit path is driven from the receive path, at the MAC internal XGMII interface. Valid values are: 0x0 = No Loopback No loopback, normal mode. 0x1 = Loopback Enabled Loopback enabled.
0	FLU	0x0	Force Link Up. 0 - Normal mode 1 - MAC is forced to the link up state regardless to the PHY link status Valid values are: 0x0 = Normal mode Normal mode 0x1 = MAC is forced to the link up state MAC is forced to the link up state regardless of the PHY link status

6.3.53.5 Data high of MACC - 0x0004

Bits	Field Name	Default NVM Value	Description
15:7	Reserved	0x0	Reserved
6	Reserved_legacy	0x0	Link down in fatal interrupt. 1 = Link indication is gated with the Fatal interrupt from the PHY. When the PHY assert fatal interrupt the link is down. 0 = Normal operation. The link indication is the same as comes from the PCS and auto negotiation machinery.
5:4	BLKXGMII	0x0	Block XGMII interface. 00 = Normal operation. The XGMII interface fed from the PCS while link is down. 01 = Idle state. The XGMII interface fed with Idles while link is down. 10 = Local Fault state. The XGMII interface fed with Local Fault sequences while link is down. 11 = Reserved Valid values are: 0x0 = Normal operation 0x1 = Idle state 0x2 = Local Fault state 0x3 = Reserved
3:0	Reserved	0x0	Reserved.

6.3.53.6 Starting Address Low at MFLCN - 0x0005

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of MFLCN	0x4294	
3:0	Type	0x1	

6.3.53.7 Starting Address High at MFLCN - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of MFLCN		

6.3.53.8 Data low of MFLCN - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	Data low of MFLCN	0x2	

6.3.53.9 Data high of MFLCN - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	Data high of MFLCN	0x0	

6.3.54 Master Reset 0 Registers Autoload Module Section Summary Table

Default setup to registers and internal memories that load on Master reset events

Word Offset	Description	Reference
0x0000	Module Length	See section 6.3.54.1
0x0001	Address Low at LEDCTL	See section 6.3.54.2
0x0002	Address High at LEDCTL	See section 6.3.54.3
0x0003	LED Control Lower Word	See section 6.3.54.4
0x0004	LED Control Upper Word	See section 6.3.54.5
0x0005	Address Low at ESDP	See section 6.3.54.6
0x0006	Address High at ESDP	See section 6.3.54.7
0x0007	SDP Control	See section 6.3.54.8
0x0008	Reserved	See section 6.3.54.9
0x0009	Address Low at WUC	See section 6.3.54.10
0x000A	Address High at WUC	See section 6.3.54.11
0x000B	Filter Control	See section 6.3.54.12
0x000C	Reserved	See section 6.3.54.13

6.3.54.1 Module Length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 1 First Section -> Word: Master Reset 0 Registers Autoload Module -> Module Length Last Section -> Word: Master Reset 0 Registers Autoload Module -> Reserved

6.3.54.2 Address Low at LEDCTL - 0x0001

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of LEDCTL	0x0200	
3:0	Type	0x1	

6.3.54.3 Address High at LEDCTL - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of LEDCTL		

6.3.54.4 LED Control Lower Word - 0x0003

The LEDCTL register defaults are loaded from two words as listed in the following tables.

Bits	Field Name	Default NVM Value	Description
15	LED1_BLINK	0x1	Valid values are: 0x0 = Don't Blink 0x1 = Blink
14	LED1_IVRT	0x0	Valid values are: 0x0 = Active Low 0x1 = Active High
13:12	Reserved	0x0	
11:8	LED1_MODE	0x1	LED 0 Control Valid values are: 0x0 = LINK_UP 0x1 = LINK 10G 0x2 = MAC ACTIVITY 0x3 = FILTER ACTIVITY 0x4 = LINK/ACTIVITY 0x5 = LINK 1G 0x6 = LINK 100M 0xE = LED ON 0xF = LED OFF
7	LED0_BLINK	0x0	Valid values are: 0x0 = Don't Blink 0x1 = Blink
6	LED0_IVRT	0x0	Valid values are: 0x0 = Active Low 0x1 = Active High
5	Global Link Mode	0x0	Valid values are: 0x0 = 200 ms 0x1 = 83 ms
4	Reserved	0x0	
3:0	LED0_MODE	0x0	LED 0 Control Valid values are: 0x0 = LINK_UP 0x1 = LINK 10G 0x2 = MAC ACTIVITY 0x3 = FILTER ACTIVITY 0x4 = LINK/ACTIVITY 0x5 = LINK 1G 0x6 = LINK 100M 0xE = LED ON 0xF = LED OFF

6.3.54.5 LED Control Upper Word - 0x0004

Bits	Field Name	Default NVM Value	Description
15	LED3_BLINK	0x0	Valid values are: 0x0 = Don't Blink 0x1 = Blink
14	LED3_IVRT	0x0	Valid values are: 0x0 = Active Low 0x1 = Active High
13:12	Reserved	0x0	
11:8	LED3_MODE	0x0	LED 0 Control Valid values are: 0x0 = LINK_UP 0x1 = LINK 10G 0x2 = MAC ACTIVITY 0x3 = FILTER ACTIVITY 0x4 = LINK/ACTIVITY 0x5 = LINK 1G 0x6 = LINK 100M 0xE = LED ON 0xF = LED OFF
7	LED2_BLINK	0x0	Valid values are: 0x0 = Don't Blink 0x1 = Blink
6	LED2_IVRT	0x0	Valid values are: 0x0 = Active Low 0x1 = Active High
5:4	Reserved	0x0	
3:0	LED2_MODE	0x0	LED 0 Control Valid values are: 0x0 = LINK_UP 0x1 = LINK 10G 0x2 = MAC ACTIVITY 0x3 = FILTER ACTIVITY 0x4 = LINK/ACTIVITY 0x5 = LINK 1G 0x6 = LINK 100M 0xE = LED ON 0xF = LED OFF

6.3.54.6 Address Low at ESDP - 0x0005

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of ESDP	0x0020	
3:0	Type	0x1	

6.3.54.7 Address High at ESDP - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of ESPD		

6.3.54.8 SDP Control - 0x0007

Bits	Field Name	Default NVM Value	Description
15	SDP3_NATIVE	0x0	Defines SDP1 operating mode is mapped to ESDP.SDP1_NATIVE loaded at power up: 0 - Operates as generic software controlled IO. 1 - Native mode operation (hardware function). Valid values are: 0x0 = Generic software controlled IO 0x1 = Native mode operation (hardware function)
14	SDP2_NATIVE	0x0	Defines SDP1 operating mode is mapped to ESDP.SDP1_NATIVE loaded at power up: 0 - Operates as generic software controlled IO. 1 - Native mode operation (hardware function). Valid values are: 0x0 = Generic software controlled IO 0x1 = Native mode operation (hardware function)
13	SDP1_NATIVE	0x0	Defines SDP1 operating mode is mapped to ESDP.SDP1_NATIVE loaded at power up: 0 - Operates as generic software controlled IO. 1 - Native mode operation (hardware function). Valid values are: 0x0 = Generic software controlled IO 0x1 = Native mode operation (hardware function)
12	SDP0_NATIVE	0x0	Defines SDP1 operating mode is mapped to ESDP.SDP1_NATIVE loaded at power up: 0 - Operates as generic software controlled IO. 1 - Native mode operation (hardware function). Valid values are: 0x0 = Generic software controlled IO 0x1 = Native mode operation (hardware function)
11	SDPDIR[3]	0x0	SDP3 Pin - Initial Direction is mapped to ESDP.SDP3_IODIR loaded at power up.
10	SDPDIR[2]	0x0	SDP2 Pin - Initial Direction is mapped to ESDP.SDP2_IODIR loaded at power up.
9	SDPDIR[1]	0x0	SDP1 Pin - Initial Direction is mapped to ESDP.SDP1_IODIR loaded at power up.
8	SDPDIR[0]	0x0	SDP0 Pin - Initial Direction is mapped to ESDP.SDP0_IODIR loaded at power up.
7	Reserved	0x0	Reserved
6	REVERSE_DIR	0x0	When set reverses the polarity of the pin direction signal to pads block. such that pads are fed with output enable signal instead of output disable.Should be cleared for normal operation.
5	SDP23_function	0x0	Defines the initial value for ESDP.SDP23_function 0 = 1588 functionality 1 = i2c functionality. Relevant only if SDP2_NATIVE or SDP3_NATIVE is set. If this bit is set, then SDP2_NATIVE and SDP3_NATIVE should have the same value.
4	SDP1_function	0x0	Defines the initial value for ESDP.SDP1_function 0b = Time Sync functionality. 1b = SDP1 Thermal sensor functionality. SDPDIR[1] should be configured as an output. Relevant only if SDP1_NATIVE is set.

Bits	Field Name	Default NVM Value	Description
3	SDPVAL[3]	0x0	SDP3 Pin - Initial Output Value is mapped to ESDP.SDP3_DATA loaded at power up.
2	SDPVAL[2]	0x0	SDP2 Pin - Initial Output Value is mapped to ESDP.SDP2_DATA loaded at power up.
1	SDPVAL[1]	0x0	SDP1 Pin - Initial Output Value is mapped to ESDP.SDP1_DATA loaded at power up.
0	SDPVAL[0]	0x0	SDP0 Pin - Initial Output Value is mapped to ESDP.SDP0_DATA loaded at power up.

6.3.54.9 Reserved - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	

6.3.54.10 Address Low at WUC - 0x0009

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of WUC	0x5800	
3:0	Type	0x1	

6.3.54.11 Address High at WUC - 0x000A

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of WUC		

6.3.54.12 Filter Control - 0x000B

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0010	Reserved

6.3.54.13 Reserved - 0x000C

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	

6.3.55 OEM Port Config 0 Section Summary Table

This section contains two sub-sections:

1. Vendor Specific Settings: Settings for external PHY or Modules. This part has a proprietary format in order to enable advanced, vendor specific, PHY setting. These settings are loaded into the external devices via the MDIO/I2C interface.
2. List of Qualified Modules: Parameter list of up to 16 modules. Per module list holds OUI, Revision and Version numbers.

Word Offset	Description	Reference
0x0000	Section Length	See section 6.3.55.1
0x0001	MNG_IF_SEL_LSW	See section 6.3.55.2
0x0002	MNG_IF_SEL_MSW	See section 6.3.55.3
0x0003	HW FW LED OPTION	See section 6.3.55.4
0x0004	SFP_LIMIT	See section 6.3.55.5
0x0005	1G_C2C_COEFF_Low	See section 6.3.55.6
0x0006	1G_C2C_COEFF_HIGH	See section 6.3.55.7
0x0007	10G_C2C_COEFF_Low	See section 6.3.55.8
0x0008	10G_C2C_COEFF_HIGH	See section 6.3.55.9
0x0009	10G_C2M_COEFF_Low	See section 6.3.55.10
0x000A	10G_C2M_COEFF_HIGH	See section 6.3.55.11
0x000B - 0x000E	Reserved (25G_C2C_COEFF_Low)	See section 6.3.55.12
0x000F	FW LED0 Port Affinity Low	See section 6.3.55.13
0x0010	FW LED0 Port Affinity High	See section 6.3.55.14
0x0011	FW LED0 Condition 0	See section 6.3.55.15
0x0012	FW LED0 Condition 1	See section 6.3.55.16
0x0013	FW LED0 Condition 2	See section 6.3.55.17
0x0014	FW LED0 Condition 3	See section 6.3.55.18
0x0015	FW LED1 Port Affinity Low	See section 6.3.55.19
0x0016	FW LED1 Port Affinity High	See section 6.3.55.20
0x0017	FW LED1 Condition 0	See section 6.3.55.21
0x0018	FW LED1 Condition 1	See section 6.3.55.22
0x0019	FW LED1 Condition 2	See section 6.3.55.23
0x001A	FW LED1 Condition 3	See section 6.3.55.24
0x001B	FW LED2 Port Affinity Low	See section 6.3.55.25
0x001C	FW LED2 Port Affinity High	See section 6.3.55.26
0x001D	FW LED2 Condition 0	See section 6.3.55.27
0x001E	FW LED2 Condition 1	See section 6.3.55.28
0x001F	FW LED2 Condition 2	See section 6.3.55.29
0x0020	FW LED2 Condition 3	See section 6.3.55.30
0x0021	FW LED3 Port Affinity Low	See section 6.3.55.31

Word Offset	Description	Reference
0x0022	FW LED3 Port Affinity High	See section 6.3.55.32
0x0023	FW LED3 Condition 0	See section 6.3.55.33
0x0024	FW LED3 Condition 1	See section 6.3.55.34
0x0025	FW LED3 Condition 2	See section 6.3.55.35
0x0026	FW LED3 Condition 3	See section 6.3.55.36

6.3.55.1 Section Length - 0x0000

The length of the section in words. Note that section length does not include a count for the section length word.

Bits	Field Name	Default NVM Value	Description
15:0	Section Length		Length in: 2 Bytes unit - 1 First Section -> Word: OEM Port Config 0 -> Section Length Last Section -> Word: OEM Port Config 0 -> FW LED3 Condition 3

6.3.55.2 MNG_IF_SEL_LSW - 0x0001

Bits	Field Name	Default NVM Value	Description
15:9	I2C_PHY_ADDR_6_0	0x0	I2C address bit[6:0]
8:4	MDIO_PHY_ADDR_4_0	0x0	MDIO addr[4:0]
3	I2C Enable	0x0	0 = I2C not Enable 1 = I2C Enable
2	MDIO Enable	0x0	0 = MDIO not Enable 1 = MDIO Enable
1	MDIO clause	0x0	0 = MDIO clause 22 1 = MDIO clause 45
0	Reserved	0x0	

6.3.55.3 MNG_IF_SEL_MSW - 0x0002

Bits	Field Name	Default NVM Value	Description
15	USXGMII Enable	0x0	0x0 = Not Enabled 0x1 = Enabled
14	SGMII Enabled	0x0	0x0 = SGMII Disabled 0x1 = SGMII Enabled
13:11	PHY Mode	0x0	000 = Backplane 001 = SFP+ 010 = Reserved (QSFP) 011 = 10GBASE-T 100 = 1GBASE-T 101 = Reserved 110 = Reserved 111 = Reserved

Bits	Field Name	Default NVM Value	Description
10:9	Netlist Selection	0x0	00 = LKV MCP netlist (1GBASE-T or 10GBASE-T) 01 = C2M (SFI/CR/CRS/CR1/SR/SX/LX) 10 = C2C (SGMII/USXGMII) 11 = Backplane (SFI/KR/KRS/KR1)
8	AN37 Enabled	0x0	0 = Disable 1 = Enable
7	AN73 Enabled	0x0	0 = Disable 1 = Enable
6:1	PHY Speed Mask	0x0	bit[5] =Reserved (25G) bit[4] =10G bit[3] =2.5G bit[2] =1G Bit[1] =100M Bit[0] =10M
0	Reserved	0x0	

6.3.55.4 HW FW LED OPTION - 0x0003

Bits	Field Name	Default NVM Value	Description
15	LED Option	0x0	0x0 = Use hardware to drive LED 0x1 = Use firmware to drive LED
14:0	Reserved	0x0	

6.3.55.5 SFP_LIMIT - 0x0004

Bits	Field Name	Default NVM Value	Description
15:12	SFP Capability Mask	0x0	Bit 15 set to 1 = Masking the SR capability Bit 14 set to 1 = Masking the LR capability Bit 13 set to 1 =Masking the SX capability Bit 12 set to 1 = masking the LX capability This bit only applies when the netlist selection filed is set to 01 (C2M)
11:0	Reserved	0x0	

6.3.55.6 1G_C2C_COEFF_Low - 0x0005

Bits	Field Name	Default NVM Value	Description
15:8	Init.C[-0]	0x0	
7:0	Init.C[1]	0x0	

6.3.55.7 1G_C2C_COEFF_HIGH - 0x0006

Bits	Field Name	Default NVM Value	Description
15	Rx Polarity	0x0	
14	Tx Polarity	0x0	
13:9	Reserved	0x0	
8	TXFFE Override	0x0	
7:0	Init.C[-1]	0x0	

6.3.55.8 10G_C2C_COEFF_Low - 0x0007

Bits	Field Name	Default NVM Value	Description
15:8	Init.C[-0]	0x0	
7:0	Init.C[1]	0x0	

6.3.55.9 10G_C2C_COEFF_HIGH - 0x0008

Bits	Field Name	Default NVM Value	Description
15:9	Reserved	0x0	
8	TXFFE Override	0x0	
7:0	Init.C[-1]	0x0	

6.3.55.10 10G_C2M_COEFF_Low - 0x0009

Bits	Field Name	Default NVM Value	Description
15:8	Init.C[-0]	0x0	
7:0	Init.C[1]	0x0	

6.3.55.11 10G_C2M_COEFF_HIGH - 0x000A

Bits	Field Name	Default NVM Value	Description
15:9	Reserved	0x0	
8	TXFFE Override	0x0	
7:0	Init.C[-1]	0x0	

6.3.55.12 Reserved (0x000B - 0x000E)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	Reserved.

6.3.55.13 FW LED0 Port Affinity Low - 0x000F

Bits	Field Name	Default NVM Value	Description
15:2	Reserved	0x0	
1:0	Port Affinity	0x0	bit 1 = Port 1 Bit 0 = Port 0

6.3.55.14 FW LED0 Port Affinity High - 0x0010

Bits	Field Name	Default NVM Value	Description
15:14	Logic	0x0	0x0 = Reserved 0x1 = AND 0x2 = OR 0x3 = Reserved
13:0	Reserved	0x0	

6.3.55.15 FW LED0 Condition 0 - 0x0011

Bits	Field Name	Default NVM Value	Description
15:14	Reserved	0x0	
13:11	Blink	0x0	0 = No blink 1 = blink 2 = blink on MAC activity others = Reserved
10:8	Color	0x0	see the Link topology section
7:6	Logic	0x0	0x0 = Reserved 0x1 = AND 0x2 = OR 0x3 = Reserved
5:0	Condition	0x0	see the Link topology section

6.3.55.16 FW LED0 Condition 1 - 0x0012

Bits	Field Name	Default NVM Value	Description
15:14	Reserved	0x0	
13:11	Blink	0x0	0 = No blink 1 = blink 2 = blink on MAC activity others = Reserved
10:8	Color	0x0	see the Link topology section
7:6	Logic	0x0	0x0 = Reserved 0x1 = AND 0x2 = OR 0x3 = Reserved
5:0	Condition	0x0	see the Link topology section

6.3.55.17 FW LED0 Condition 2 - 0x0013

Bits	Field Name	Default NVM Value	Description
15:14	Reserved	0x0	
13:11	Blink	0x0	0 = No blink 1 = blink 2 = blink on MAC activity others = Reserved
10:8	Color	0x0	see the Link topology section
7:6	Logic	0x0	0x0 = Reserved 0x1 = AND 0x2 = OR 0x3 = Reserved
5:0	Condition	0x0	see the Link topology section

6.3.55.18 FW LED0 Condition 3 - 0x0014

Bits	Field Name	Default NVM Value	Description
15:14	Reserved	0x0	
13:11	Blink	0x0	0 = No blink 1 = blink 2 = blink on MAC activity others = Reserved
10:8	Color	0x0	see the Link topology section
7:6	Logic	0x0	0x0 = Reserved 0x1 = AND 0x2 = OR 0x3 = Reserved
5:0	Condition	0x0	see the Link topology section

6.3.55.19 FW LED1 Port Affinity Low - 0x0015

Bits	Field Name	Default NVM Value	Description
15:2	Reserved	0x0	
1:0	Port Affinity	0x0	Bit 1 = Port 1 Bit 0 = Port 0

6.3.55.20 FW LED1 Port Affinity High - 0x0016

Bits	Field Name	Default NVM Value	Description
15:14	Logic	0x0	0x0 = Reserved 0x1 = AND 0x2 = OR 0x3 = Reserved
13:0	Reserved	0x0	

6.3.55.21 FW LED1 Condition 0 - 0x0017

Bits	Field Name	Default NVM Value	Description
15:14	Reserved	0x0	
13:11	Blink	0x0	0= No blink 1 = blink 2 = blink on MAC activity others = Reserved
10:8	Color	0x0	see the Link topology section
7:6	Logic	0x0	0x0 = Reserved 0x1 =AND 0x2 = OR 0x3 = Reserved
5:0	Condition	0x0	see the Link topology section

6.3.55.22 FW LED1 Condition 1 - 0x0018

Bits	Field Name	Default NVM Value	Description
15:14	Reserved	0x0	
13:11	Blink	0x0	0= No blink 1 = blink 2 = blink on MAC activity others = Reserved
10:8	Color	0x0	see the Link topology section
7:6	Logic	0x0	0x0 = Reserved 0x1 =AND 0x2 = OR 0x3 = Reserved
5:0	Condition	0x0	see the Link topology section

6.3.55.23 FW LED1 Condition 2 - 0x0019

Bits	Field Name	Default NVM Value	Description
15:14	Reserved	0x0	
13:11	Blink	0x0	0= No blink 1 = blink 2 = blink on MAC activity others = Reserved
10:8	Color	0x0	see the Link topology section
7:6	Logic	0x0	0x0 = Reserved 0x1 =AND 0x2 = OR 0x3 = Reserved
5:0	Condition	0x0	see the Link topology section

6.3.55.24 FW LED1 Condition 3 - 0x001A

Bits	Field Name	Default NVM Value	Description
15:14	Reserved	0x0	
13:11	Blink	0x0	0 = No blink 1 = blink 2 = blink on MAC activity others = Reserved
10:8	Color	0x0	see the Link topology section
7:6	Logic	0x0	0x0 = Reserved 0x1 = AND 0x2 = OR 0x3 = Reserved
5:0	Condition	0x0	see the Link topology section

6.3.55.25 FW LED2 Port Affinity Low - 0x001B

Bits	Field Name	Default NVM Value	Description
15:2	Reserved	0x0	
1:0	Port Affinity	0x0	Bit 1 = Port 1 Bit 0 = Port 0

6.3.55.26 FW LED2 Port Affinity High - 0x001C

Bits	Field Name	Default NVM Value	Description
15:14	Logic	0x0	0x0 = Reserved 0x1 = AND 0x2 = OR 0x3 = Reserved
13:0	Reserved	0x0	

6.3.55.27 FW LED2 Condition 0 - 0x001D

Bits	Field Name	Default NVM Value	Description
15:14	Reserved	0x0	
13:11	Blink	0x0	0 = No blink 1 = blink 2 = blink on MAC activity others = Reserved
10:8	Color	0x0	see the Link topology section
7:6	Logic	0x0	0x0 = Reserved 0x1 = AND 0x2 = OR 0x3 = Reserved
5:0	Condition	0x0	see the Link topology section

6.3.55.28 FW LED2 Condition 1 - 0x001E

Bits	Field Name	Default NVM Value	Description
15:14	Reserved	0x0	
13:11	Blink	0x0	0= No blink 1 = blink 2 = blink on MAC activity others = Reserved
10:8	Color	0x0	see the Link topology section
7:6	Logic	0x0	0x0 = Reserved 0x1 =AND 0x2 = OR 0x3 = Reserved
5:0	Condition	0x0	see the Link topology section

6.3.55.29 FW LED2 Condition 2 - 0x001F

Bits	Field Name	Default NVM Value	Description
15:14	Reserved	0x0	
13:11	Blink	0x0	0= No blink 1 = blink 2 = blink on MAC activity others = Reserved
10:8	Color	0x0	see the Link topology section
7:6	Logic	0x0	0x0 = Reserved 0x1 =AND 0x2 = OR 0x3 = Reserved
5:0	Condition	0x0	see the Link topology section

6.3.55.30 FW LED2 Condition 3 - 0x0020

Bits	Field Name	Default NVM Value	Description
15:14	Reserved	0x0	
13:11	Blink	0x0	0= No blink 1 = blink 2 = blink on MAC activity others = Reserved
10:8	Color	0x0	see the Link topology section
7:6	Logic	0x0	0x0 = Reserved 0x1 =AND 0x2 = OR 0x3 = Reserved
5:0	Condition	0x0	see the Link topology section

6.3.55.31 FW LED3 Port Affinity Low - 0x0021

Bits	Field Name	Default NVM Value	Description
15:2	Reserved	0x0	
1:0	Port Affinity	0x0	Bit 1 = Port 1 Bit 0 = Port 0

6.3.55.32 FW LED3 Port Affinity High - 0x0022

Bits	Field Name	Default NVM Value	Description
15:14	Logic	0x0	0x0 = Reserved 0x1 = AND 0x2 = OR 0x3 = Reserved
13:0	Reserved	0x0	

6.3.55.33 FW LED3 Condition 0 - 0x0023

Bits	Field Name	Default NVM Value	Description
15:14	Reserved	0x0	
13:11	Blink	0x0	0 = No blink 1 = blink 2 = blink on MAC activity others = Reserved
10:8	Color	0x0	see the Link topology section
7:6	Logic	0x0	0x0 = Reserved 0x1 = AND 0x2 = OR 0x3 = Reserved
5:0	Condition	0x0	see the Link topology section

6.3.55.34 FW LED3 Condition 1 - 0x0024

Bits	Field Name	Default NVM Value	Description
15:14	Reserved	0x0	
13:11	Blink	0x0	0 = No blink 1 = blink 2 = blink on MAC activity others = Reserved
10:8	Color	0x0	see the Link topology section
7:6	Logic	0x0	0x0 = Reserved 0x1 = AND 0x2 = OR 0x3 = Reserved
5:0	Condition	0x0	see the Link topology section

6.3.55.35 FW LED3 Condition 2 - 0x0025

Bits	Field Name	Default NVM Value	Description
15:14	Reserved	0x0	
13:11	Blink	0x0	0= No blink 1 = blink 2 = blink on MAC activity others = Reserved
10:8	Color	0x0	see the Link topology section
7:6	Logic	0x0	0x0 = Reserved 0x1 = AND 0x2 = OR 0x3 = Reserved
5:0	Condition	0x0	see the Link topology section

6.3.55.36 FW LED3 Condition 3 - 0x0026

Bits	Field Name	Default NVM Value	Description
15:14	Reserved	0x0	
13:11	Blink	0x0	0= No blink 1 = blink 2 = blink on MAC activity others = Reserved
10:8	Color	0x0	see the Link topology section
7:6	Logic	0x0	0x0 = Reserved 0x1 = AND 0x2 = OR 0x3 = Reserved
5:0	Condition	0x0	see the Link topology section

6.3.56 NVM Image CSS Header Section Summary Table

Word Offset	Description	Reference
0x0000	moduleTypeL	See section 6.3.56.1
0x0001	moduleTypeH	See section 6.3.56.2
0x0002	headerLenL	See section 6.3.56.3
0x0003	headerLenH	See section 6.3.56.4
0x0004	headerVersionL	See section 6.3.56.5
0x0005	headerVersionH	See section 6.3.56.6
0x0006	moduleIDL	See section 6.3.56.7
0x0007	moduleIDH	See section 6.3.56.8
0x0008	moduleVendorL	See section 6.3.56.9
0x0009	moduleVendorH	See section 6.3.56.10
0x000A	dateL	See section 6.3.56.11
0x000B	dateH	See section 6.3.56.12
0x000C	sizeL	See section 6.3.56.13
0x000D	sizeH	See section 6.3.56.14
0x000E	keySizeL	See section 6.3.56.15
0x000F	keySizeH	See section 6.3.56.16
0x0010	modulusSizeL	See section 6.3.56.17
0x0011	modulusSizeH	See section 6.3.56.18
0x0012	exponentSizeL	See section 6.3.56.19
0x0013	exponentSizeH	See section 6.3.56.20
0x0014	lad_srevL	See section 6.3.56.21
0x0015	lad_srevH	See section 6.3.56.22
0x0016	crypto_infoL	See section 6.3.56.23
0x0017	crypto_infoH	See section 6.3.56.24
0x0018	lad_fw_entry_offsetL	See section 6.3.56.25
0x0019	lad_fw_entry_offsetH	See section 6.3.56.26
0x001A	reserved2L	See section 6.3.56.27
0x001B	reserved2H	See section 6.3.56.28
0x001C	lad_image_unique_idL	See section 6.3.56.29
0x001D	lad_image_unique_idH	See section 6.3.56.30
0x001E	lad_module_idL	See section 6.3.56.31
0x001F	lad_module_idH	See section 6.3.56.32
0x0020	reserved	See section 6.3.56.33

6.3.56.1 moduleTypeL - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	moduleType	0x6	

6.3.56.2 moduleTypeH - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	moduleTypeH		

6.3.56.3 headerLenL - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	headerLenL	0x1A1	

6.3.56.4 headerLenH - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	headerLenH		

6.3.56.5 headerVersionL - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionL	0x00010000	

6.3.56.6 headerVersionH - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionH		

6.3.56.7 moduleIDL - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	moduleIDL	0x0	

6.3.56.8 moduleIDH - 0x0007

Bits	Field Name	Default NVM Value	Description
15	signMode	0x1	
14:0	moduleIDH		

6.3.56.9 moduleVendorL - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorL	0x00008086	

6.3.56.10 moduleVendorH - 0x0009

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorH		

6.3.56.11 dateL - 0x000A

0xMMDD

Bits	Field Name	Default NVM Value	Description
15:0	DateL	0x20130530	0xMMDD

6.3.56.12 dateH - 0x000B

0xYYYY

Bits	Field Name	Default NVM Value	Description
15:0	DateH		

6.3.56.13 sizeL - 0x000C

size is in DWords

Bits	Field Name	Default NVM Value	Description
15:0	sizeL	0x106800	

6.3.56.14 sizeH - 0x000D

size is in DWords

Bits	Field Name	Default NVM Value	Description
15:0	sizeH		

6.3.56.15 keySizeL - 0x000E

Bits	Field Name	Default NVM Value	Description
15:0	keySizeL	0x60	

6.3.56.16 keySizeH - 0x000F

Bits	Field Name	Default NVM Value	Description
15:0	keySizeH		

6.3.56.17 modulusSizeL - 0x0010

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeL	0x60	

6.3.56.18 modulusSizeH - 0x0011

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeH		

6.3.56.19 exponentSizeL - 0x0012

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeL	0x1	

6.3.56.20 exponentSizeH - 0x0013

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeH		

6.3.56.21 lad_srevL - 0x0014

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevL	0x1	

6.3.56.22 lad_srevH - 0x0015

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevH		

6.3.56.23 crypto_infoL - 0x0016

Bits	Field Name	Default NVM Value	Description
15:8	hash algorithm	0x1	
7:0	sign mode	0x8	

6.3.56.24 crypto_infoH - 0x0017

Bits	Field Name	Default NVM Value	Description
15:12	reserved	0x0	
11:8	srev_flags	0xC	
7:0	encryption algorithm	0x0	

6.3.56.25 lad_fw_entry_offsetL - 0x0018

offset of NVM image from the beginning of NVM CSS module

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetL	0x106C0	

6.3.56.26 lad_fw_entry_offsetH - 0x0019

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetH		

6.3.56.27 reserved2L - 0x001A

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.56.28 reserved2H - 0x001B

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.56.29 lad_image_unique_idL - 0x001C

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idL		

6.3.56.30 lad_image_unique_idH - 0x001D

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idH		

6.3.56.31 lad_module_idL - 0x001E

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idL	0x6	Valid values are: 0x1 = EMP Image 0x2 = PE Image 0x3 = PCIe Analog 0x4 = PHY Analog 0x5 = Option ROM 0x6 = NVM Bank 0x7 = Extended Miniloader

6.3.56.32 lad_module_idH - 0x001F

Bits	Field Name	Default NVM Value	Description
15	Reserved	0x0	
14:0	lad_module_idH		

6.3.56.33 reserved - 0x0020

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.57 NVM Key and Signature Section Summary Table

Word Offset	Description	Reference
0x0000 + 1*n, n=0...191	RSA Public Key	See section 6.3.57.1
0x00C0 + 1*n, n=0...191	RSA Public Key Prime	See section 6.3.57.2
0x0180 + 1*n, n=0...191	RSA Public Key Second	See section 6.3.57.3
0x0240	RSA ExponentL	See section 6.3.57.4
0x0241	RSA ExponentH	See section 6.3.57.5
0x0242 + 1*n, n=0...191	Encrypted SHA384 Hash	See section 6.3.57.6

6.3.57.1 RSA Public Key[n] (0x0000 + 1*n, n=0...191)

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.3.57.2 RSA Public Key Prime[n] (0x00C0 + 1*n, n=0...191)

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key Prime		

6.3.57.3 RSA Public Key Second[n] (0x0180 + 1*n, n=0...191)

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key Second		

6.3.57.4 RSA ExponentL - 0x0240

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentL	0x0	

6.3.57.5 RSA ExponentH - 0x0241

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentH	0x0	

6.3.57.6 Encrypted SHA384 Hash[n] (0x0242 + 1*n, n=0...191)

Bits	Field Name	Default NVM Value	Description
15:0	Encrypted SHA384 Hash	0x0	

6.3.58 NVM Image Auth Header Section Summary Table

Word Offset	Description	Reference
0x0000	Device Blank NVM Device ID	See section 6.3.58.1
0x0001	Max Module AreaL	See section 6.3.58.2
0x0002	Max Module AreaH	See section 6.3.58.3
0x0003	Current Module AreaL	See section 6.3.58.4
0x0004	Current Module AreaH	See section 6.3.58.5
0x0005	Reserved	See section 6.3.58.6
0x0006	Code Revision	See section 6.3.58.7
0x0007	Reserved Spare Word	See section 6.3.58.8

6.3.58.1 Device Blank NVM Device ID - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Device Blank NVM Device ID	0x57AC	

6.3.58.2 Max Module AreaL - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaL	0x1000	

6.3.58.3 Max Module AreaH - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaH	0x0009	

6.3.58.4 Current Module AreaL - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaL	0xE000	

6.3.58.5 Current Module AreaH - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaH	0x0006	

6.3.58.6 Reserved - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	

6.3.58.7 Code Revision - 0x0006

Bits	Field Name	Default NVM Value	Description
15:8	Major Revision	0x0	
7:0	Minor Revision	0x0	

6.3.58.8 Reserved Spare Word - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	Reserved Spare Word	0x0	

6.3.59 Padding section Section Summary Table

Padding section in order to keep the SR copy at address 0x10160 for images with 2K key

Word Offset	Description	Reference
0x0000	Padding	See section 6.3.59.1

6.3.59.1 Padding - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Padding	0xFF	

6.3.60 SR1 - should be copy of Shadow RAM: Section clone

SR1 - should be copy of Shadow RAM is a clone of data starting at section Init Module word NVM Control Word 1 and ending at section Last Word of Shadow RAM word Last Word of Shadow RAM

6.3.61 ML CSS Header Section Summary Table

Word Offset	Description	Reference
0x0000	moduleTypeL	See section 6.3.61.1
0x0001	moduleTypeH	See section 6.3.61.2
0x0002	headerLenL	See section 6.3.61.3
0x0003	headerLenH	See section 6.3.61.4
0x0004	headerVersionL	See section 6.3.61.5
0x0005	headerVersionH	See section 6.3.61.6
0x0006	moduleIDL	See section 6.3.61.7
0x0007	moduleIDH	See section 6.3.61.8
0x0008	moduleVendorL	See section 6.3.61.9
0x0009	moduleVendorH	See section 6.3.61.10
0x000A	dateL	See section 6.3.61.11
0x000B	dateH	See section 6.3.61.12
0x000C	sizeL	See section 6.3.61.13
0x000D	sizeH	See section 6.3.61.14
0x000E	keySizeL	See section 6.3.61.15
0x000F	keySizeH	See section 6.3.61.16
0x0010	modulusSizeL	See section 6.3.61.17
0x0011	modulusSizeH	See section 6.3.61.18
0x0012	exponentSizeL	See section 6.3.61.19
0x0013	exponentSizeH	See section 6.3.61.20
0x0014	lad_srevL	See section 6.3.61.21
0x0015	lad_srevH	See section 6.3.61.22
0x0016	crypto_infoL	See section 6.3.61.23
0x0017	crypto_infoH	See section 6.3.61.24
0x0018	lad_fw_entry_offsetL	See section 6.3.61.25
0x0019	lad_fw_entry_offsetH	See section 6.3.61.26
0x001A	reserved2L	See section 6.3.61.27
0x001B	reserved2H	See section 6.3.61.28
0x001C	lad_image_unique_idL	See section 6.3.61.29
0x001D	lad_image_unique_idH	See section 6.3.61.30
0x001E	lad_module_idL	See section 6.3.61.31
0x001F	lad_module_idH	See section 6.3.61.32
0x0020 + 1*n, n=0...31	reserved	See section 6.3.61.33

6.3.61.1 moduleTypeL - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	moduleType	0x6	

6.3.61.2 moduleTypeH - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	moduleTypeH		

6.3.61.3 headerLenL - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	headerLenL	0x1a1	

6.3.61.4 headerLenH - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	headerLenH		

6.3.61.5 headerVersionL - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionL	0x00010000	

6.3.61.6 headerVersionH - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionH		

6.3.61.7 moduleIDL - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	moduleIDL	0x0	

6.3.61.8 moduleIDH - 0x0007

Bits	Field Name	Default NVM Value	Description
15	signMode	0x0	
14:0	moduleIDH		

6.3.61.9 moduleVendorL - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorL	0x00008086	

6.3.61.10 moduleVendorH - 0x0009

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorH		

6.3.61.11 dateL - 0x000A

0xMMDD

Bits	Field Name	Default NVM Value	Description
15:0	DateL	0x20130530	0xMMDD

6.3.61.12 dateH - 0x000B

0xYYYY

Bits	Field Name	Default NVM Value	Description
15:0	DateH		

6.3.61.13 sizeL - 0x000C

Bits	Field Name	Default NVM Value	Description
15:0	sizeL	0x8000	

6.3.61.14 sizeH - 0x000D

Bits	Field Name	Default NVM Value	Description
15:0	sizeH		

6.3.61.15 keySizeL - 0x000E

Bits	Field Name	Default NVM Value	Description
15:0	keySizeL	0x60	

6.3.61.16 keySizeH - 0x000F

Bits	Field Name	Default NVM Value	Description
15:0	keySizeH		

6.3.61.17 modulusSizeL - 0x0010

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeL	0x60	

6.3.61.18 modulusSizeH - 0x0011

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeH		

6.3.61.19 exponentSizeL - 0x0012

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeL	0x1	

6.3.61.20 exponentSizeH - 0x0013

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeH		

6.3.61.21 lad_srevL - 0x0014

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevL	0x1	

6.3.61.22 lad_srevH - 0x0015

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevH		

6.3.61.23 crypto_infoL - 0x0016

Bits	Field Name	Default NVM Value	Description
15:8	hash algorithm	0x1	
7:0	sign mode	0x8	

6.3.61.24 crypto_infoH - 0x0017

Bits	Field Name	Default NVM Value	Description
15:12	reserved	0x0	
11:8	srev_flags	0xC	
7:0	encryption algorithm	0x0	

6.3.61.25 lad_fw_entry_offsetL - 0x0018

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetL	0x6C4	

6.3.61.26 lad_fw_entry_offsetH - 0x0019

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetH		

6.3.61.27 reserved2L - 0x001A

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.61.28 reserved2H - 0x001B

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.61.29 lad_image_unique_idL - 0x001C

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idL	0x0	

6.3.61.30 lad_image_unique_idH - 0x001D

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idH	0x0	

6.3.61.31 lad_module_idL - 0x001E

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idL	0x7	Valid values are: 0x1 = EMP Image 0x2 = PE Image 0x3 = PCIe Analog 0x4 = PHY Analog 0x5 = Option ROM 0x6 = NVM Bank 0x7 = Extended Miniloader

6.3.61.32 lad_module_idH - 0x001F

Bits	Field Name	Default NVM Value	Description
15	Reserved	0x0	
14:0	lad_module_idH		

6.3.61.33 reserved[n] (0x0020 + 1*n, n=0...31)

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.62 ML Key and Signature Section Summary Table

Word Offset	Description	Reference
0x0000 + 1*n, n=0...191	RSA Public Key	See section 6.3.62.1
0x00C0 + 1*n, n=0...191	RSA Public Key Prime	See section 6.3.62.2
0x0180 + 1*n, n=0...191	RSA Public Key Second	See section 6.3.62.3
0x0240	RSA ExponentL	See section 6.3.62.4
0x0241	RSA ExponentH	See section 6.3.62.5
0x0242 + 1*n, n=0...191	Encrypted SHA384 Hash	See section 6.3.62.6

6.3.62.1 RSA Public Key[n] (0x0000 + 1*n, n=0...191)

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.3.62.2 RSA Public Key Prime[n] (0x00C0 + 1*n, n=0...191)

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key Prime		

6.3.62.3 RSA Public Key Second[n] (0x0180 + 1*n, n=0...191)

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key Second		

6.3.62.4 RSA ExponentL - 0x0240

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentL	0x0	

6.3.62.5 RSA ExponentH - 0x0241

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentH	0x0	

6.3.62.6 Encrypted SHA384 Hash[n] (0x0242 + 1*n, n=0...191)

Bits	Field Name	Default NVM Value	Description
15:0	Encrypted SHA384 Hash	0x0	

6.3.63 ML Auth Header Section Summary Table

Word Offset	Description	Reference
0x0000	Device Blank NVM Device ID	See section 6.3.63.1
0x0001	Max Module AreaL	See section 6.3.63.2
0x0002	Max Module AreaH	See section 6.3.63.3
0x0003	Current Module AreaL	See section 6.3.63.4
0x0004	Current Module AreaH	See section 6.3.63.5
0x0005	Reserved	See section 6.3.63.6
0x0006	Code Revision	See section 6.3.63.7
0x0007	Reserved Spare Word	See section 6.3.63.8

6.3.63.1 Device Blank NVM Device ID - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Device Blank NVM Device ID	0x57AC	

6.3.63.2 Max Module AreaL - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaL	0x1000	

6.3.63.3 Max Module AreaH - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaH	0x0009	

6.3.63.4 Current Module AreaL - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaL	0xE000	

6.3.63.5 Current Module AreaH - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaH	0x0006	

6.3.63.6 Reserved - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	

6.3.63.7 Code Revision - 0x0006

Bits	Field Name	Default NVM Value	Description
15:8	Major Revision	0x0	
7:0	Minor Revision	0x0	

6.3.63.8 Reserved Spare Word - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	Reserved Spare Word	0x0	

6.3.64 Extended ML Header Section Summary Table

Word Offset	Description	Reference
0x0000	Reserved	See section 6.3.64.1
0x0001	Analog PHY pre PLL configuration pointer	See section 6.3.64.2
0x0002	CSR protected list pointer	See section 6.3.64.3
0x0003	PCIe Analog pointer	See section 6.3.64.4
0x0004	PCIR fixed autoload pointer	See section 6.3.64.5
0x0005	POR fixed autoload pointer	See section 6.3.64.6
0x0006	PCIR PFA autoload allowlist pointer	See section 6.3.64.7
0x0007	POR PFA autoload allowlist pointer	See section 6.3.64.8
0x0008	Reserved	See section 6.3.64.9
0x0009	1st NVM Bank pointer	See section 6.3.64.10
0x000A	NVM Bank Area Size	See section 6.3.64.11
0x000B	1st OROM Bank pointer	See section 6.3.64.12
0x000C	OROM Bank Area Size	See section 6.3.64.13
0x000D	1st TLV extension bank pointer	See section 6.3.64.14
0x000E	TLV Extension Bank Area Size	See section 6.3.64.15
0x000F	Preserved Field Area pointer	See section 6.3.64.16
0x0010	Recovery Firmware Pointer	See section 6.3.64.17
0x0011	Reserved	See section 6.3.64.18
0x0012	Factory Settings Size	See section 6.3.64.19
0x0013	Last PFA word Pointer	See section 6.3.64.20
0x0014	MAC Reset autoload pointer	See section 6.3.64.21
0x0015	Reserved	See section 6.3.64.22
0x0016	PCIe Config Group 1 Hash Low	See section 6.3.64.23
0x0017	PCIe Config Group 1 Hash High	See section 6.3.64.24
0x0018	Master Reset autoload pointer	See section 6.3.64.25

6.3.64.1 Reserved - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Reserved		Reserved.

6.3.64.2 Analog PHY pre PLL configuration pointer - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Analog PLL configuration pointer	0x0	

Field Analog PLL configuration pointer points to Analog PHY pre PLL Configuration Section. For Analog PHY pre PLL Configuration inner structure See [section 6.3.66](#)

6.3.64.3 CSR protected list pointer - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	CSR protected list pointer	0x0	

Field CSR protected list pointer points to CSR Protected List Section. For CSR Protected List inner structure See [section 6.3.67](#)

6.3.64.4 PCIe Analog pointer - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	PCIe Analog pointer	0x0	

Field PCIe Analog pointer points to PCIe Analog Module Section. For PCIe Analog Module inner structure See [section 6.3.68](#)

6.3.64.5 PCIR fixed autoload pointer - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	PCIR fixed autoload pointer	0x0	Relative pointer

Field PCIR fixed autoload pointer points to PCIR Registers PFA Autoload Module Section. For PCIR Registers PFA Autoload Module inner structure See [section 6.3.69](#)

6.3.64.6 POR fixed autoload pointer - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	POR fixed autoload pointer	0x0	Relative pointer

Field POR fixed autoload pointer points to POR Registers Autoload Module Section. For POR Registers Autoload Module inner structure See [section 6.3.70](#)

6.3.64.7 PCIR PFA autoload allowlist pointer - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	PCIR PFA autoload allowlist pointer	0x0	Relative pointer

Field PCIR PFA autoload allowlist pointer points to PCIR_PFA Autoload Allowlist Module Section. For PCIR_PFA Autoload Allowlist Module inner structure See [section 6.3.71](#)

6.3.64.8 POR PFA autoload allowlist pointer - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	POR PFA autoload allowlist pointer	0x0	Relative pointer

Field POR PFA autoload allowlist pointer points to POR_PFA Autoload Allowlist Module Section. For POR_PFA Autoload Allowlist Module inner structure See [section 6.3.72](#)

6.3.64.9 Reserved - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	Reserved.

6.3.64.10 1st NVM Bank pointer - 0x0009

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	1st NVM Bank pointer	0x020	Pointer to first NVM section (not swapped when section changes)

6.3.64.11 NVM Bank Area Size - 0x000A

Bits	Field Name	Default NVM Value	Description
15:12	RESERVED	0x0	
11:0	NVM Bank Area Size	0x41A	Size expressed in 4 KB sector units.

6.3.64.12 1st OROM Bank pointer - 0x000B

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	1st OROM Bank pointer	0x0854	Pointer to first OROM section (not swapped when section changes) Note: This value is fixed to 4224 KB - need to be manually changed if location of section changes.

6.3.64.13 OROM Bank Area Size - 0x000C

Bits	Field Name	Default NVM Value	Description
15:12	RESERVED	0x0	
11:0	OROM Bank Area Size	0x7D	Size expressed in 4 KB sector units.

6.3.64.14 1st TLV extension bank pointer - 0x000D

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	1st TLV extension bank pointer	0x094E	Pointer to first TLV extension section (not swapped when section changes) Note: This value is fixed to 4224 KB - need to be manually changed if location of section changes.

6.3.64.15 TLV Extension Bank Area Size - 0x000E

Bits	Field Name	Default NVM Value	Description
15:12	RESERVED	0x0	
11:0	TLV Extension Bank Area Size	0x07	Size expressed in 4 KB sector units.

6.3.64.16 Preserved Field Area pointer - 0x000F

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	PFA pointer	0x100	Pointer to PFA in first NVM section (not swapped when section changes).

6.3.64.17 Recovery Firmware Pointer - 0x0010

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	Recovery Firmware Pointer	0x053	Absolute pointer to Recovery Firmware section.

6.3.64.18 Reserved- 0x0011

Bits	Field Name	Default NVM Value	Description
15:0	Reserved		Reserved.

6.3.64.19 Factory Settings Size - 0x0012

Bits	Field Name	Default NVM Value	Description
15:12	RESERVED	0x00	
11:0	Factory Settings Size	0xD	Size expressed in 4 KB sector units.

6.3.64.20 Last PFA word Pointer - 0x0013

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x0	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	Last PFA word Pointer	0x2FFF	Pointer to the last word of the PFA in first NVM section (not swapped when section changes).

6.3.64.21 MAC Reset autoload pointer - 0x0014

Bits	Field Name	Default NVM Value	Description
15:0	MAC Reset autoload pointer	0x0	Relative pointer.

Field MAC Reset autoload pointer points to MAC Reset Registers Autoload Module Section. For MAC Reset Registers Autoload Module inner structure See [section 6.3.73](#)

6.3.64.22 Reserved - 0x0015

Bits	Field Name	Default NVM Value	Description
15:0	Reserved		Relative pointer.

6.3.64.23 PCIe Config Group 1 Hash Low - 0x0016

Bits	Field Name	Default NVM Value	Description
15:0	PCIe Config Group 1 Hash Low		CRC-32: Start Section -> Word: PCIe Analog Module -> Module Length End Section -> Word: POR_PFA Autoload Allowlist Module -> Data High of GL_MNG_HWARB_CTRL

6.3.64.24 PCIe Config Group 1 Hash High - 0x0017

Bits	Field Name	Default NVM Value	Description
15:0	PCIe Config Group 1 Hash High		

6.3.64.25 Master Reset autoload pointer - 0x0018

Bits	Field Name	Default NVM Value	Description
15:0	Master Reset autoload pointer	0x0	Relative pointer.

Field Master Reset autoload pointer points to Master Reset Registers Autoload Module Section. For Master Reset Registers Autoload Module inner structure See [section 6.3.74](#)

6.3.65 ML Image Section Summary Table

This module is the whole module which has been signed by CSS.
It is in the NVM map for EM tool needs only,

6.3.66 Analog PHY pre PLL Configuration Section Summary Table

Includes static configurations of PHY registers for PLL lock

Word Offset	Description	Reference
0x0000	Section Length	See section 6.3.66.1
0x0001	Reg Write Indirect List	See section 6.3.66.2

6.3.66.1 Section Length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Section Length		Length in: 2 Bytes unit - 1 First Section -> Word: Analog PHY pre PLL Configuration -> Section Length Last Section -> Word: Analog PHY pre PLL Configuration -> Reg Write Indirect List

6.3.66.2 Reg Write Indirect List - 0x0001

Raw data module length: variable

6.3.67 CSR Protected List Section Summary Table

Defines the list of the protected CSRs and their default settings. These registers are made RO to the host as they contains settings which are critical for the host to flash access when in blank flash programming mode.

Word Offset	Description	Reference
0x0000	Module Length	See section 6.3.67.1
0x0001 - 0x0008	Reserved	See section 6.3.67.2
0x0009	Address Low at GLGEN_STAT	See section 6.3.67.3
0x000A	Address High at GLGEN_STAT	See section 6.3.67.4
0x000B	Data Low of GLGEN_STAT	See section 6.3.67.5
0x000C	Data High of GLGEN_STAT	See section 6.3.67.6
0x000D	Address Low at GLNVM_ALTIMERS	See section 6.3.67.7
0x000E	Address High at GLNVM_ALTIMERS	See section 6.3.67.8
0x000F	Data Low of GLNVM_ALTIMERS	See section 6.3.67.9
0x0010	Data High of GLNVM_ALTIMERS	See section 6.3.67.10
0x0011 - 0x0014	Reserved	See section 6.3.67.11
0x0015	Starting Address Low at GLNVM_PROTCSR[0]	See section 6.3.67.12
0x0016	Starting Address High at GLNVM_PROTCSR[0]	See section 6.3.67.13
0x0017	Attributes at GLNVM_PROTCSR[0]	See section 6.3.67.14
0x0018	Data Low of GLNVM_PROTCSR[0]	See section 6.3.67.15
0x0019	Data High of GLNVM_PROTCSR[0]	See section 6.3.67.16
0x001A	Data Low of GLNVM_PROTCSR[1]	See section 6.3.67.17
0x001B	Data High of GLNVM_PROTCSR[1]	See section 6.3.67.18
0x001C	Data Low of GLNVM_PROTCSR[2]	See section 6.3.67.19
0x001D	Data High of GLNVM_PROTCSR[2]	See section 6.3.67.20
0x001E	Data Low of GLNVM_PROTCSR[3]	See section 6.3.67.21
0x001F	Data High of GLNVM_PROTCSR[3]	See section 6.3.67.22
0x0020	Data Low of GLNVM_PROTCSR[4]	See section 6.3.67.23
0x0021	Data High of GLNVM_PROTCSR[4]	See section 6.3.67.24
0x0022	Data Low of GLNVM_PROTCSR[5]	See section 6.3.67.25
0x0023	Data High of GLNVM_PROTCSR[5]	See section 6.3.67.26
0x0024	Data Low of GLNVM_PROTCSR[6]	See section 6.3.67.27
0x0025	Data High of GLNVM_PROTCSR[6]	See section 6.3.67.28
0x0026	Data Low of GLNVM_PROTCSR[7]	See section 6.3.67.29
0x0027	Data High of GLNVM_PROTCSR[7]	See section 6.3.67.30
0x0028	Data Low of GLNVM_PROTCSR[8]	See section 6.3.67.31
0x0029	Data High of GLNVM_PROTCSR[8]	See section 6.3.67.32
0x002A	Data Low of GLNVM_PROTCSR[9]	See section 6.3.67.33
0x002B	Data High of GLNVM_PROTCSR[9]	See section 6.3.67.34

Word Offset	Description	Reference
0x002C	Data Low of GLNVM_PROTCSR[10]	See section 6.3.67.35
0x002D	Data High of GLNVM_PROTCSR[10]	See section 6.3.67.36
0x002E	Data Low of GLNVM_PROTCSR[11]	See section 6.3.67.37
0x002F	Data High of GLNVM_PROTCSR[11]	See section 6.3.67.38
0x0030	Data Low of GLNVM_PROTCSR[12]	See section 6.3.67.39
0x0031	Data High of GLNVM_PROTCSR[12]	See section 6.3.67.40
0x0032	Data Low of GLNVM_PROTCSR[13]	See section 6.3.67.41
0x0033	Data High of GLNVM_PROTCSR[13]	See section 6.3.67.42
0x0034	Data Low of GLNVM_PROTCSR[14]	See section 6.3.67.43
0x0035	Data High of GLNVM_PROTCSR[14]	See section 6.3.67.44
0x0036	Data Low of GLNVM_PROTCSR[15]	See section 6.3.67.45
0x0037	Data High of GLNVM_PROTCSR[15]	See section 6.3.67.46
0x0038	Data Low of GLNVM_PROTCSR[16]	See section 6.3.67.47
0x0039	Data High of GLNVM_PROTCSR[16]	See section 6.3.67.48
0x003A	Data Low of GLNVM_PROTCSR[17]	See section 6.3.67.49
0x003B	Data High of GLNVM_PROTCSR[17]	See section 6.3.67.50
0x003C	Data Low of GLNVM_PROTCSR[18]	See section 6.3.67.51
0x003D	Data High of GLNVM_PROTCSR[18]	See section 6.3.67.52
0x003E	Data Low of GLNVM_PROTCSR[19]	See section 6.3.67.53
0x003F	Data High of GLNVM_PROTCSR[19]	See section 6.3.67.54
0x0040	Data Low of GLNVM_PROTCSR[20]	See section 6.3.67.55
0x0041	Data High of GLNVM_PROTCSR[20]	See section 6.3.67.56
0x0042	Data Low of GLNVM_PROTCSR[21]	See section 6.3.67.57
0x0043	Data High of GLNVM_PROTCSR[21]	See section 6.3.67.58
0x0044	Data Low of GLNVM_PROTCSR[22]	See section 6.3.67.59
0x0045	Data High of GLNVM_PROTCSR[22]	See section 6.3.67.60
0x0046	Data Low of GLNVM_PROTCSR[23]	See section 6.3.67.61
0x0047	Data High of GLNVM_PROTCSR[23]	See section 6.3.67.62
0x0048	Data Low of GLNVM_PROTCSR[24]	See section 6.3.67.63
0x0049	Data High of GLNVM_PROTCSR[24]	See section 6.3.67.64
0x004A	Data Low of GLNVM_PROTCSR[25]	See section 6.3.67.65
0x004B	Data High of GLNVM_PROTCSR[25]	See section 6.3.67.66
0x004C	Data Low of GLNVM_PROTCSR[26]	See section 6.3.67.67
0x004D	Data High of GLNVM_PROTCSR[26]	See section 6.3.67.68
0x004E	Data Low of GLNVM_PROTCSR[27]	See section 6.3.67.69
0x004F	Data High of GLNVM_PROTCSR[27]	See section 6.3.67.70
0x0050	Data Low of GLNVM_PROTCSR[28]	See section 6.3.67.71
0x0051	Data High of GLNVM_PROTCSR[28]	See section 6.3.67.72
0x0052	Data Low of GLNVM_PROTCSR[29]	See section 6.3.67.73

Word Offset	Description	Reference
0x0053	Data High of GLNVM_PROTCSR[29]	See section 6.3.67.74
0x0054	Data Low of GLNVM_PROTCSR[30]	See section 6.3.67.75
0x0055	Data High of GLNVM_PROTCSR[30]	See section 6.3.67.76
0x0056	Data Low of GLNVM_PROTCSR[31]	See section 6.3.67.77
0x0057	Data High of GLNVM_PROTCSR[31]	See section 6.3.67.78
0x0058	Data Low of GLNVM_PROTCSR[32]	See section 6.3.67.79
0x0059	Data High of GLNVM_PROTCSR[32]	See section 6.3.67.80
0x005A	Data Low of GLNVM_PROTCSR[33]	See section 6.3.67.81
0x005B	Data High of GLNVM_PROTCSR[33]	See section 6.3.67.82
0x005C	Data Low of GLNVM_PROTCSR[34]	See section 6.3.67.83
0x005D	Data High of GLNVM_PROTCSR[34]	See section 6.3.67.84
0x005E	Data Low of GLNVM_PROTCSR[35]	See section 6.3.67.85
0x005F	Data High of GLNVM_PROTCSR[35]	See section 6.3.67.86
0x0060	Data Low of GLNVM_PROTCSR[36]	See section 6.3.67.87
0x0061	Data High of GLNVM_PROTCSR[36]	See section 6.3.67.88
0x0062	Data Low of GLNVM_PROTCSR[37]	See section 6.3.67.89
0x0063	Data High of GLNVM_PROTCSR[37]	See section 6.3.67.90
0x0064	Data Low of GLNVM_PROTCSR[38]	See section 6.3.67.91
0x0065	Data High of GLNVM_PROTCSR[38]	See section 6.3.67.92
0x0066	Data Low of GLNVM_PROTCSR[39]	See section 6.3.67.93
0x0067	Data High of GLNVM_PROTCSR[39]	See section 6.3.67.94
0x0068	Data Low of GLNVM_PROTCSR[40]	See section 6.3.67.95
0x0069	Data High of GLNVM_PROTCSR[40]	See section 6.3.67.96
0x006A	Data Low of GLNVM_PROTCSR[41]	See section 6.3.67.97
0x006B	Data High of GLNVM_PROTCSR[41]	See section 6.3.67.98
0x006C	Data Low of GLNVM_PROTCSR[42]	See section 6.3.67.99
0x006D	Data High of GLNVM_PROTCSR[42]	See section 6.3.67.100
0x006E	Data Low of GLNVM_PROTCSR[43]	See section 6.3.67.101
0x006F	Data High of GLNVM_PROTCSR[43]	See section 6.3.67.102
0x0070	Data Low of GLNVM_PROTCSR[44]	See section 6.3.67.103
0x0071	Data High of GLNVM_PROTCSR[44]	See section 6.3.67.104
0x0072	Data Low of GLNVM_PROTCSR[45]	See section 6.3.67.105
0x0073	Data High of GLNVM_PROTCSR[45]	See section 6.3.67.106
0x0074	Data Low of GLNVM_PROTCSR[46]	See section 6.3.67.107
0x0075	Data High of GLNVM_PROTCSR[46]	See section 6.3.67.108
0x0076	Data Low of GLNVM_PROTCSR[47]	See section 6.3.67.109
0x0077	Data High of GLNVM_PROTCSR[47]	See section 6.3.67.110
0x0078	Data Low of GLNVM_PROTCSR[48]	See section 6.3.67.111
0x0079	Data High of GLNVM_PROTCSR[48]	See section 6.3.67.112

Word Offset	Description	Reference
0x007A	Data Low of GLNVM_PROTCSR[49]	See section 6.3.67.113
0x007B	Data High of GLNVM_PROTCSR[49]	See section 6.3.67.114
0x007C	Data Low of GLNVM_PROTCSR[50]	See section 6.3.67.115
0x007D	Data High of GLNVM_PROTCSR[50]	See section 6.3.67.116
0x007E	Data Low of GLNVM_PROTCSR[51]	See section 6.3.67.117
0x007F	Data High of GLNVM_PROTCSR[51]	See section 6.3.67.118
0x0080	Data Low of GLNVM_PROTCSR[52]	See section 6.3.67.119
0x0081	Data High of GLNVM_PROTCSR[52]	See section 6.3.67.120
0x0082	Data Low of GLNVM_PROTCSR[53]	See section 6.3.67.121
0x0083	Data High of GLNVM_PROTCSR[53]	See section 6.3.67.122
0x0084	Data Low of GLNVM_PROTCSR[54]	See section 6.3.67.123
0x0085	Data High of GLNVM_PROTCSR[54]	See section 6.3.67.124
0x0086	Data Low of GLNVM_PROTCSR[55]	See section 6.3.67.125
0x0087	Data High of GLNVM_PROTCSR[55]	See section 6.3.67.126
0x0088	Data Low of GLNVM_PROTCSR[56]	See section 6.3.67.127
0x0089	Data High of GLNVM_PROTCSR[56]	See section 6.3.67.128
0x008A	Data Low of GLNVM_PROTCSR[57]	See section 6.3.67.129
0x008B	Data High of GLNVM_PROTCSR[57]	See section 6.3.67.130
0x008C	Data Low of GLNVM_PROTCSR[58]	See section 6.3.67.131
0x008D	Data High of GLNVM_PROTCSR[58]	See section 6.3.67.132
0x008E	Data Low of GLNVM_PROTCSR[59]	See section 6.3.67.133
0x008F	Data High of GLNVM_PROTCSR[59]	See section 6.3.67.134

6.3.67.1 Module Length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 1 First Section -> Word: CSR Protected List -> Module Length Last Section -> Word: CSR Protected List -> Data High of GLNVM_PROTCSR[59]

6.3.67.2 Reserved (0x0001 - 0x0008)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved		Reserved.

6.3.67.3 Address Low at GLGEN_STAT - 0x0009

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLGEN_STAT	0xB612C	
3:0	Type	0x1	

6.3.67.4 Address High at GLGEN_STAT - 0x000A

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLGEN_STAT		

6.3.67.5 Data Low of GLGEN_STAT - 0x000B

Bits	Field Name	Default NVM Value	Description
15:0	RESERVED	0x0	Reserved.

6.3.67.6 Data High of GLGEN_STAT - 0x000C

Bits	Field Name	Default NVM Value	Description
15:0	RESERVED cont		

6.3.67.7 Address Low at GLNVM_ALTIMERS - 0x000D

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLNVM_ALTIMERS	0xB6140	
3:0	Type	0x1	

6.3.67.8 Address High at GLNVM_ALTIMERS - 0x000E

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLNVM_ALTIMERS		

6.3.67.9 Data Low of GLNVM_ALTIMERS - 0x000F

Bits	Field Name	Default NVM Value	Description
15:12	GEN_ALTIMER	0x1BE	General Autoload Timer. Default is ~ 300 ms. resolution is in ms.
11:0	PCI_ALTIMER	0x16	PCIe Autoload Timer. Default is 16 to 15 ms. resolution is in ms.

6.3.67.10 Data High of GLNVM_ALTIMERS - 0x0010

Bits	Field Name	Default NVM Value	Description
15:0	GEN_ALTIMER cont		

6.3.67.11 Reserved (0x0011 - 0x0014)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved		Reserved.

6.3.67.12 Starting Address Low at GLNVM_PROTCSR[0] - 0x0015

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GLNVM_PROTCSR	0xB610	
3:0	Type	0x2	

6.3.67.13 Starting Address High at GLNVM_PROTCSR[0] - 0x0016

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GLNVM_PROTCSR		

6.3.67.14 Attributes at GLNVM_PROTCSR[0] - 0x0017

Bits	Field Name	Default NVM Value	Description
15:5	Length	0x3C	
4:3	Skip	0x0	
2:0	Width	0x0	

6.3.67.15 Data Low of GLNVM_PROTCSR[0] - 0x0018

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x824C4	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.16 Data High of GLNVM_PROTCSR[0] - 0x0019

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.17 Data Low of GLNVM_PROTCSR[1] - 0x001A

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x83054	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.18 Data High of GLNVM_PROTCSR[1] - 0x001B

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.19 Data Low of GLNVM_PROTCSR[2] - 0x001C

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x8305C	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.20 Data High of GLNVM_PROTCSR[2] - 0x001D

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.21 Data Low of GLNVM_PROTCSR[3] - 0x001E

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x8306C	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.22 Data High of GLNVM_PROTCSR[3] - 0x001F

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.23 Data Low of GLNVM_PROTCSR[4] - 0x0020

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x83200	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.24 Data High of GLNVM_PROTCSR[4] - 0x0021

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.25 Data Low of GLNVM_PROTCSR[5] - 0x0022

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x83204	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.26 Data High of GLNVM_PROTCSR[5] - 0x0023

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.27 Data Low of GLNVM_PROTCSR[6] - 0x0024

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x83208	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.28 Data High of GLNVM_PROTCSR[6] - 0x0025

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.29 Data Low of GLNVM_PROTCSR[7] - 0x0026

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x83240	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.30 Data High of GLNVM_PROTCSR[7] - 0x0027

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.31 Data Low of GLNVM_PROTCSR[8] - 0x0028

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x83244	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.32 Data High of GLNVM_PROTCSR[8] - 0x0029

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.33 Data Low of GLNVM_PROTCSR[9] - 0x002A

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x83248	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.34 Data High of GLNVM_PROTCSR[9] - 0x002B

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.35 Data Low of GLNVM_PROTCSR[10] - 0x002C

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x8324C	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.36 Data High of GLNVM_PROTCSR[10] - 0x002D

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.37 Data Low of GLNVM_PROTCSR[11] - 0x002E

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x9DECC	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.38 Data High of GLNVM_PROTCSR[11] - 0x002F

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.39 Data Low of GLNVM_PROTCSR[12] - 0x0030

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x9DED4	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.40 Data High of GLNVM_PROTCSR[12] - 0x0031

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.41 Data Low of GLNVM_PROTCSR[13] - 0x0032

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x9E940	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.42 Data High of GLNVM_PROTCSR[13] - 0x0033

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.43 Data Low of GLNVM_PROTCSR[14] - 0x0034

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x9E944	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.44 Data High of GLNVM_PROTCSR[14] - 0x0035

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.45 Data Low of GLNVM_PROTCSR[15] - 0x0036

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0xA4000	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.46 Data High of GLNVM_PROTCSR[15] - 0x0037

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.47 Data Low of GLNVM_PROTCSR[16] - 0x0038

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0xA4004	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.48 Data High of GLNVM_PROTCSR[16] - 0x0039

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.49 Data Low of GLNVM_PROTCSR[17] - 0x003A

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0xB6008	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.50 Data High of GLNVM_PROTCSR[17] - 0x003B

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.51 Data Low of GLNVM_PROTCSR[18] - 0x003C

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0xB6100	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.52 Data High of GLNVM_PROTCSR[18] - 0x003D

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.53 Data Low of GLNVM_PROTCSR[19] - 0x003E

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0xB6108	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.54 Data High of GLNVM_PROTCSR[19] - 0x003F

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.55 Data Low of GLNVM_PROTCSR[20] - 0x0040

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0xB6118	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.56 Data High of GLNVM_PROTCSR[20] - 0x0041

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.57 Data Low of GLNVM_PROTCSR[21] - 0x0042

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0xB612C	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.58 Data High of GLNVM_PROTCSR[21] - 0x0043

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.59 Data Low of GLNVM_PROTCSR[22] - 0x0044

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0xB6140	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.60 Data High of GLNVM_PROTCSR[22] - 0x0045

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.61 Data Low of GLNVM_PROTCSR[23] - 0x0046

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0xB6820	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.62 Data High of GLNVM_PROTCSR[23] - 0x0047

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.63 Data Low of GLNVM_PROTCSR[24] - 0x0048

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0xB7008	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.64 Data High of GLNVM_PROTCSR[24] - 0x0049

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.65 Data Low of GLNVM_PROTCSR[25] - 0x004A

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0xB7020	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.66 Data High of GLNVM_PROTCSR[25] - 0x004B

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.67 Data Low of GLNVM_PROTCSR[26] - 0x004C

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x1D2698	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.68 Data High of GLNVM_PROTCSR[26] - 0x004D

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.69 Data Low of GLNVM_PROTCSR[27] - 0x004E

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x2040A8	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.70 Data High of GLNVM_PROTCSR[27] - 0x004F

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.71 Data Low of GLNVM_PROTCSR[28] - 0x0050

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x39380C	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.72 Data High of GLNVM_PROTCSR[28] - 0x0051

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.73 Data Low of GLNVM_PROTCSR[29] - 0x0052

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.74 Data High of GLNVM_PROTCSR[29] - 0x0053

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.75 Data Low of GLNVM_PROTCSR[30] - 0x0054

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.76 Data High of GLNVM_PROTCSR[30] - 0x0055

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.77 Data Low of GLNVM_PROTCSR[31] - 0x0056

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.78 Data High of GLNVM_PROTCSR[31] - 0x0057

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.79 Data Low of GLNVM_PROTCSR[32] - 0x0058

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.80 Data High of GLNVM_PROTCSR[32] - 0x0059

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.81 Data Low of GLNVM_PROTCSR[33] - 0x005A

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.82 Data High of GLNVM_PROTCSR[33] - 0x005B

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.83 Data Low of GLNVM_PROTCSR[34] - 0x005C

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.84 Data High of GLNVM_PROTCSR[34] - 0x005D

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.85 Data Low of GLNVM_PROTCSR[35] - 0x005E

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.86 Data High of GLNVM_PROTCSR[35] - 0x005F

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.87 Data Low of GLNVM_PROTCSR[36] - 0x0060

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.88 Data High of GLNVM_PROTCSR[36] - 0x0061

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.89 Data Low of GLNVM_PROTCSR[37] - 0x0062

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.90 Data High of GLNVM_PROTCSR[37] - 0x0063

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.91 Data Low of GLNVM_PROTCSR[38] - 0x0064

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.92 Data High of GLNVM_PROTCSR[38] - 0x0065

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.93 Data Low of GLNVM_PROTCSR[39] - 0x0066

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.94 Data High of GLNVM_PROTCSR[39] - 0x0067

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.95 Data Low of GLNVM_PROTCSR[40] - 0x0068

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.96 Data High of GLNVM_PROTCSR[40] - 0x0069

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.97 Data Low of GLNVM_PROTCSR[41] - 0x006A

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.98 Data High of GLNVM_PROTCSR[41] - 0x006B

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.99 Data Low of GLNVM_PROTCSR[42] - 0x006C

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.100 Data High of GLNVM_PROTCSR[42] - 0x006D

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.101 Data Low of GLNVM_PROTCSR[43] - 0x006E

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.102 Data High of GLNVM_PROTCSR[43] - 0x006F

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.103 Data Low of GLNVM_PROTCSR[44] - 0x0070

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.104 Data High of GLNVM_PROTCSR[44] - 0x0071

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.105 Data Low of GLNVM_PROTCSR[45] - 0x0072

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.106 Data High of GLNVM_PROTCSR[45] - 0x0073

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.107 Data Low of GLNVM_PROTCSR[46] - 0x0074

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.108 Data High of GLNVM_PROTCSR[46] - 0x0075

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.109 Data Low of GLNVM_PROTCSR[47] - 0x0076

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.110 Data High of GLNVM_PROTCSR[47] - 0x0077

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.111 Data Low of GLNVM_PROTCSR[48] - 0x0078

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.112 Data High of GLNVM_PROTCSR[48] - 0x0079

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.113 Data Low of GLNVM_PROTCSR[49] - 0x007A

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.114 Data High of GLNVM_PROTCSR[49] - 0x007B

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.115 Data Low of GLNVM_PROTCSR[50] - 0x007C

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.116 Data High of GLNVM_PROTCSR[50] - 0x007D

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.117 Data Low of GLNVM_PROTCSR[51] - 0x007E

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.118 Data High of GLNVM_PROTCSR[51] - 0x007F

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.119 Data Low of GLNVM_PROTCSR[52] - 0x0080

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.120 Data High of GLNVM_PROTCSR[52] - 0x0081

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.121 Data Low of GLNVM_PROTCSR[53] - 0x0082

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.122 Data High of GLNVM_PROTCSR[53] - 0x0083

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.123 Data Low of GLNVM_PROTCSR[54] - 0x0084

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.124 Data High of GLNVM_PROTCSR[54] - 0x0085

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.125 Data Low of GLNVM_PROTCSR[55] - 0x0086

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.126 Data High of GLNVM_PROTCSR[55] - 0x0087

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.127 Data Low of GLNVM_PROTCSR[56] - 0x0088

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.128 Data High of GLNVM_PROTCSR[56] - 0x0089

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.129 Data Low of GLNVM_PROTCSR[57] - 0x008A

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.130 Data High of GLNVM_PROTCSR[57] - 0x008B

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.131 Data Low of GLNVM_PROTCSR[58] - 0x008C

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.132 Data High of GLNVM_PROTCSR[58] - 0x008D

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.67.133 Data Low of GLNVM_PROTCSR[59] - 0x008E

Bits	Field Name	Default NVM Value	Description
15:0	ADDR_BLOCK	0x0	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.

6.3.67.134 Data High of GLNVM_PROTCSR[59] - 0x008F

Bits	Field Name	Default NVM Value	Description
15:8	RESERVED	0x0	Reserved
7:0	ADDR_BLOCK cont		

6.3.68 PCIe Analog Module Section Summary Table

Contains Read-Only parameters that configure the PCIe Transaction layer

Word Offset	Description	Reference
0x0000	Module Length	See section 6.3.68.1
0x0001	PCIe Analog Data	See section 6.3.68.2

6.3.68.1 Module Length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		

6.3.68.2 PCIe Analog Data - 0x0001

Raw data module length: variable

This word shall be disabled at the image level.

6.3.69 PCIR Registers PFA Autoload Module Section Summary Table

Default setup to registers and internal memories that load on PCIR events

Word Offset	Description	Reference
0x0000	Module Length	See section 6.3.69.1
0x0001	Address Low at PCI_PWRDATA	See section 6.3.69.2
0x0002	Address High at PCI_PWRDATA	See section 6.3.69.3
0x0003	PCI_PWRDATA L	See section 6.3.69.4
0x0004	PCI_PWRDATA H	See section 6.3.69.5
0x0005	Address Low at PCI_REVID	See section 6.3.69.6
0x0006	Address High at PCI_REVID	See section 6.3.69.7
0x0007	PCI_REVID L	See section 6.3.69.8
0x0008	PCI_REVID H	See section 6.3.69.9
0x0009	Address Low at PCI_PMSUP	See section 6.3.69.10
0x000A	Address High at PCI_PMSUP	See section 6.3.69.11
0x000B	PCI_PMSUP L	See section 6.3.69.12
0x000C	PCI_PMSUP H	See section 6.3.69.13
0x000D	Address Low at PCI_PCIERR	See section 6.3.69.14
0x000E	Address High at PCI_PCIERR	See section 6.3.69.15
0x000F	PCI_PCIERR L	See section 6.3.69.16
0x0010	PCI_PCIERR H	See section 6.3.69.17

6.3.69.1 Module Length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 1 First Section -> Word: PCIR Registers PFA Autoload Module -> Module Length Last Section -> Word: PCIR Registers PFA Autoload Module -> PCI_PCIERR H

6.3.69.2 Address Low at PCI_PWRDATA - 0x0001

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_PWRDATA	0x11060	
3:0	Type	0x1	

6.3.69.3 Address High at PCI_PWRDATA - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_PWRDATA		

6.3.69.4 PCI_PWRDATA L - 0x0003

PCIe Power Data Register

Bits	Field Name	Default NVM Value	Description
15:0	PCI_PWRDATA L	0x0	

6.3.69.5 PCI_PWRDATA H - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	PCI_PWRDATA H		

6.3.69.6 Address Low at PCI_REVID - 0x0005

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_REVID	0x11098	
3:0	Type	0x1	

6.3.69.7 Address High at PCI_REVID - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_REVID		

6.3.69.8 PCI_REVID L - 0x0007

Revision ID

Bits	Field Name	Default NVM Value	Description
15:0	PCI_REVID L	0x0	

6.3.69.9 PCI_REVID H - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	PCI_REVID H		

6.3.69.10 Address Low at PCI_PMSUP - 0x0009

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_PMSUP	0x110A0	
3:0	Type	0x1	

6.3.69.11 Address High at PCI_PMSUP - 0x000A

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_PMSUP		

6.3.69.12 PCI_PMSUP L - 0x000B

PCIe Power Management Support

Bits	Field Name	Default NVM Value	Description
15:0	PCI_PMSUP L	0x00007397	

6.3.69.13 PCI_PMSUP H - 0x000C

Bits	Field Name	Default NVM Value	Description
15:0	PCI_PMSUP H		

6.3.69.14 Address Low at PCI_PCIERR - 0x000D

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_PCIERR	0x11140	
3:0	Type	0x1	

6.3.69.15 Address High at PCI_PCIERR - 0x000E

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_PCIERR		

6.3.69.16 PCI_PCIERR L - 0x000F

Error Enables

Bits	Field Name	Default NVM Value	Description
15:0	PCI_PCIERR L	0xE4FC	

6.3.69.17 PCI_PCIERR H - 0x0010

Bits	Field Name	Default NVM Value	Description
15:0	PCI_PCIERR H	0x000B	

6.3.70 POR Registers Autoload Module Section Summary Table

Default setup to registers that load on POR events

Word Offset	Description	Reference
0x0000	Module Length	See section 6.3.70.1
0x0001 - 0x001A	Starting Address Low at TAR_ACC_APRVD_SAI_LIST_0	See section 6.3.70.2

6.3.70.1 Module Length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 1 First Section -> Word: POR Registers Autoload Module -> Module Length Last Section -> Word: POR Registers Autoload Module -> Data High of SB_IOSF_RP_TYPE_H

6.3.70.2 Reserved (0x0001 - 0x001A)

Bits	Field Name	Default NVM Value	Description
15:0	Reserved		Reserved.

6.3.71 PCIR_PFA Autoload Allowlist Module Section Summary Table

Default setup to registers and internal memories that load on PCIR events

Word Offset	Description	Reference
0x0000	Module Length	See section 6.3.71.1
0x0001	Address Low at PCI_CNF2	See section 6.3.71.2
0x0002	Address High at PCI_CNF2	See section 6.3.71.3
0x0003	PCI_CNF2 low	See section 6.3.71.4
0x0004	PCI_CNF2 high	See section 6.3.71.5
0x0005	Address Low at PCI_LBARCTRL	See section 6.3.71.6
0x0006	Address High at PCI_LBARCTRL	See section 6.3.71.7
0x0007	PCI_LBARCTRL L	See section 6.3.71.8
0x0008	PCI_LBARCTRL H	See section 6.3.71.9
0x0009	Address Low at PCI_SERL	See section 6.3.71.10
0x000A	Address High at PCI_SERL	See section 6.3.71.11
0x000B	PCI_SERL 1	See section 6.3.71.12
0x000C	PCI_SERL 2	See section 6.3.71.13
0x000D	Address Low at PCI_SERH	See section 6.3.71.14
0x000E	Address High at PCI_SERH	See section 6.3.71.15
0x000F	PCI_SERL 3	See section 6.3.71.16
0x0010	PCI_SERL 4	See section 6.3.71.17
0x0011	Address Low at PCI_CAPCTRL	See section 6.3.71.18
0x0012	Address High at PCI_CAPCTRL	See section 6.3.71.19
0x0013	PCI_CAPCTRL L	See section 6.3.71.20
0x0014	PCI_CAPCTRL H	See section 6.3.71.21
0x0015	Address Low at PCI_CAPSUP	See section 6.3.71.22
0x0016	Address High at PCI_CAPSUP	See section 6.3.71.23
0x0017	PCI_CAPSUP L	See section 6.3.71.24
0x0018	PCI_CAPSUP H	See section 6.3.71.25
0x0019	Address Low at PCI_DBGCTL	See section 6.3.71.26
0x001A	Address High at PCI_DBGCTL	See section 6.3.71.27
0x001B	PCI_DBGCTL L	See section 6.3.71.28
0x001C	PCI_DBGCTL H	See section 6.3.71.29
0x001D	Address Low at PCI_UPADD	See section 6.3.71.30
0x001E	Address High at PCI_UPADD	See section 6.3.71.31
0x001F	PCI_UPADD L	See section 6.3.71.32
0x0020	PCI_UPADD H	See section 6.3.71.33
0x0021	Address Low at PCI_SUBSYSID	See section 6.3.71.34

Word Offset	Description	Reference
0x0022	Address High at PCI_SUBSYSID	See section 6.3.71.35
0x0023	PCI_SUBSYSID L	See section 6.3.71.36
0x0024	PCI_SUBSYSID H	See section 6.3.71.37
0x0025	Address Low at PCI_VFSUP	See section 6.3.71.38
0x0026	Address High at PCI_VFSUP	See section 6.3.71.39
0x0027	PCI_VFSUP L	See section 6.3.71.40
0x0028	PCI_VFSUP H	See section 6.3.71.41
0x0029	Address Low at TPH_CTRL	See section 6.3.71.42
0x002A	Address High at TPH_CTRL	See section 6.3.71.43
0x002B	TPH_CTRL L	See section 6.3.71.44
0x002C	TPH_CTRL H	See section 6.3.71.45
0x002D	Address Low at PCI_LINKCAP	See section 6.3.71.46
0x002E	Address High at PCI_LINKCAP	See section 6.3.71.47
0x002F	PCI_LINKCAP L	See section 6.3.71.48
0x0030	PCI_LINKCAP H	See section 6.3.71.49
0x0031	Address Low at PCI_GLBL_CNF	See section 6.3.71.50
0x0032	Address High at PCI_GLBL_CNF	See section 6.3.71.51
0x0033	PCI_GLBL_CNF L	See section 6.3.71.52
0x0034	PCI_GLBL_CNF H	See section 6.3.71.53
0x0035	Address Low at PCI_VENDORID	See section 6.3.71.54
0x0036	Address High at PCI_VENDORID	See section 6.3.71.55
0x0037	PCI_VENDORID L	See section 6.3.71.56
0x0038	PCI_VENDORID H	See section 6.3.71.57
0x0039	Address Low at PCI_PFDEVID	See section 6.3.71.58
0x003A	Address High at PCI_PFDEVID	See section 6.3.71.59
0x003B	PCI_PFDEVID Low (LAN)	See section 6.3.71.60
0x003C	PCI_PFDEVID High (SAN)	See section 6.3.71.61
0x003D	Address Low at PCI_CNF	See section 6.3.71.62
0x003E	Address High at PCI_CNF	See section 6.3.71.63
0x003F	PCI_CNF Low	See section 6.3.71.64
0x0040	PCI_CNF High	See section 6.3.71.65
0x0041	Address Low at PCI_VFDEVID	See section 6.3.71.66
0x0042	Address High at PCI_VFDEVID	See section 6.3.71.67
0x0043	PCI_VFDEVID Low (LAN)	See section 6.3.71.68
0x0044	PCI_VFDEVID High (SAN)	See section 6.3.71.69
0x0045	Address Low at PCI_CLASS	See section 6.3.71.70
0x0046	Address High at PCI_CLASS	See section 6.3.71.71
0x0047	PCI_CLASS Low	See section 6.3.71.72
0x0048	PCI_CLASS High	See section 6.3.71.73

Word Offset	Description	Reference
0x0049	Address Low at PCI_FUNC2	See section 6.3.71.74
0x004A	Address High at PCI_FUNC2	See section 6.3.71.75
0x004B	PCI_FUNC2 Low	See section 6.3.71.76
0x004C	PCI_FUNC2 High	See section 6.3.71.77
0x004D	Address Low at PCI_PFDEVID	See section 6.3.71.78
0x004E	Address High at PCI_PFDEVID	See section 6.3.71.79
0x004F	PCI_PFDEVID Low (LAN)	See section 6.3.71.80
0x0050	PCI_PFDEVID High (SAN)	See section 6.3.71.81
0x0051	Address Low at PCI_CNF	See section 6.3.71.82
0x0052	Address High at PCI_CNF	See section 6.3.71.83
0x0053	PCI_CNF Low	See section 6.3.71.84
0x0054	PCI_CNF High	See section 6.3.71.85
0x0055	Address Low at PCI_VFDEVID	See section 6.3.71.86
0x0056	Address High at PCI_VFDEVID	See section 6.3.71.87
0x0057	PCI_VFDEVID Low (LAN)	See section 6.3.71.88
0x0058	PCI_VFDEVID High (SAN)	See section 6.3.71.89
0x0059	Address Low at PCI_CLASS	See section 6.3.71.90
0x005A	Address High at PCI_CLASS	See section 6.3.71.91
0x005B	PCI_CLASS Low	See section 6.3.71.92
0x005C	PCI_CLASS High	See section 6.3.71.93
0x005D	Address Low at PCI_FUNC2	See section 6.3.71.94
0x005E	Address High at PCI_FUNC2	See section 6.3.71.95
0x005F	PCI_FUNC2 Low	See section 6.3.71.96
0x0060	PCI_FUNC2 High	See section 6.3.71.97

6.3.71.1 Module Length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 1 First Section -> Word: PCIR_PFA Autoload Allowlist Module -> Module Length Last Section -> Word: PCIR_PFA Autoload Allowlist Module -> PCI_FUNC2 High

6.3.71.2 Address Low at PCI_CNF2 - 0x0001

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_CNF2	0x110F8	
3:0	Type	0x1	

6.3.71.3 Address High at PCI_CNF2 - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_CNF2		

6.3.71.4 PCI_CNF2 low - 0x0003

MSI-x and cache line configuration

Bits	Field Name	Default NVM Value	Description
15:0	PCI_CNF2 low	0x400040FC	

6.3.71.5 PCI_CNF2 high - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	PCI_CNF2 high		

6.3.71.6 Address Low at PCI_LBARCTRL - 0x0005

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_LBARCTRL	0x11048	
3:0	Type	0x1	

6.3.71.7 Address High at PCI_LBARCTRL - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_LBARCTRL		

6.3.71.8 PCI_LBARCTRL L - 0x0007

PF BAR registers configuration

Bits	Field Name	Default NVM Value	Description
15:14	RESERVED	0x0	
13:11	EXROM_BAR_SIZE	0x3	
10:9	RESERVED	0x0	
8:6	FL_BAR_SIZE	0x7	This field indicates the size of the external Flash as = 64KB x (2 ** FL_BAR_SIZE) as used to define exposure in the BAR. The following values are supported: 101b - 2MB 110b - 4MB 111b - 8MB Else - Reserved
5:4	RESERVED	0x0	
3	RESERVED	0x1	
2	CSRSIZE	0x1	The CSRSIZE and FL_BAR_SIZE fields define the usable FLASH size and CSR mapping window size.
1	BAR32	0x0	BAR 32bit Enable. 0b - 64bit BAR addressing mode is selected. 1b - 32bit BARs are enabled.
0	PREFBAR	0x1	Prefetchable bit indication in the memory BARs (should be set when 64bit BARs are used) 0b - BARs are marked as non prefetchable 1b - BARs are marked as prefetchable

6.3.71.9 PCI_LBARCTRL H - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	PCI_LBARCTRL H	0x0	

6.3.71.10 Address Low at PCI_SERL - 0x0009

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_SERL	0x110F0	
3:0	Type	0x1	

6.3.71.11 Address High at PCI_SERL - 0x000A

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_SERL		

6.3.71.12 PCI_SERL 1 - 0x000B

Serial MAC Address Byte0 [LSB] and Byte1

Bits	Field Name	Default NVM Value	Description
15:0	PCI_SERL 1	0x000	Serial MAC Address Byte0 [LSB] and Byte1 Note: This field is preserved by Intel NVM update tool.

6.3.71.13 PCI_SERL 2 - 0x000C

Serial MAC Address Byte2 [LSB] and Byte3

Bits	Field Name	Default NVM Value	Description
15:0	PCI_SERL 2	0xc9	Serial MAC Address Byte2 [LSB] and Byte3 Note: This field is preserved by Intel NVM update tool.

6.3.71.14 Address Low at PCI_SERH - 0x000D

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_SERH	0x11078	
3:0	Type	0x1	

6.3.71.15 Address High at PCI_SERH - 0x000E

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_SERH		

6.3.71.16 PCI_SERL 3 - 0x000F

Serial MAC Address Byte4 [LSB] and Byte5

Bits	Field Name	Default NVM Value	Description
15:0	PCI_SERL 3	0x0	Serial MAC Address Byte4 [LSB] and Byte5 Note: This field is preserved by Intel NVM update tool.

6.3.71.17 PCI_SERL 4 - 0x0010

Bits	Field Name	Default NVM Value	Description
15:0	PCI_SERL 4	0x0	Note: This field is preserved by Intel NVM update tool.

6.3.71.18 Address Low at PCI_CAPCTRL - 0x0011

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_CAPCTRL	0x11080	
3:0	Type	0x1	

6.3.71.19 Address High at PCI_CAPCTRL - 0x0012

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_CAPCTRL		

6.3.71.20 PCI_CAPCTRL L - 0x0013

VPD_EN

Bits	Field Name	Default NVM Value	Description
15:1	PCI_CAPCTRL L	0x0	
0	VPD_EN	0x1	

6.3.71.21 PCI_CAPCTRL H - 0x0014

Bits	Field Name	Default NVM Value	Description
15:0	PCI_CAPCTRL H	0x0	

6.3.71.22 Address Low at PCI_CAPSUP - 0x0015

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_CAPSUP	0x11088	
3:0	Type	0x1	

6.3.71.23 Address High at PCI_CAPSUP - 0x0016

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_CAPSUP		

6.3.71.24 PCI_CAPSUP L - 0x0017

PCIe Capabilities exposed

Bits	Field Name	Default NVM Value	Description
15:0	PCI_CAPSUP L	0x1C4EFF	

6.3.71.25 PCI_CAPSUP H - 0x0018

Bits	Field Name	Default NVM Value	Description
15:0	PCI_CAPSUP H		

6.3.71.26 Address Low at PCI_DBGCTL - 0x0019

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_DBGCTL	0x110E4	
3:0	Type	0x1	

6.3.71.27 Address High at PCI_DBGCTL - 0x001A

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_DBGCTL		

6.3.71.28 PCI_DBGCTL L - 0x001B

Allow access to VF config space

Bits	Field Name	Default NVM Value	Description
15:0	PCI_DBGCTL L	0x1	

6.3.71.29 PCI_DBGCTL H - 0x001C

Bits	Field Name	Default NVM Value	Description
15:0	PCI_DBGCTL H		

6.3.71.30 Address Low at PCI_UPADD - 0x001D

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_UPADD	0x110E8	
3:0	Type	0x1	

6.3.71.31 Address High at PCI_UPADD - 0x001E

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_UPADD		

6.3.71.32 PCI_UPADD L - 0x001F

Upper Address accessible

Bits	Field Name	Default NVM Value	Description
15:1	ADDRESS	0x0	
0	MODE	0x0	

6.3.71.33 PCI_UPADD H - 0x0020

Bits	Field Name	Default NVM Value	Description
15:0	ADDRESS		

6.3.71.34 Address Low at PCI_SUBSYSID - 0x0021

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_SUBSYSID	0x11058	
3:0	Type	0x1	

6.3.71.35 Address High at PCI_SUBSYSID - 0x0022

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_SUBSYSID		

6.3.71.36 PCI_SUBSYSID L - 0x0023

Sub-Vendor ID

Bits	Field Name	Default NVM Value	Description
15:0	SUB_VEN_ID	0x8086	Note: This field is preserved by Intel NVM update tool.

6.3.71.37 PCI_SUBSYSID H - 0x0024

Sub-System ID

Bits	Field Name	Default NVM Value	Description
15:0	SUB_ID	0x0	Note: This field is preserved by Intel NVM update tool.

6.3.71.38 Address Low at PCI_VFSUP - 0x0025

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_VFSUP	0x110A8	
3:0	Type	0x1	

6.3.71.39 Address High at PCI_VFSUP - 0x0026

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_VFSUP		

6.3.71.40 PCI_VFSUP L - 0x0027

PF BAR registers configuration

Bits	Field Name	Default NVM Value	Description
15:0	PCI_VFSUP L	0x0	

6.3.71.41 PCI_VFSUP H - 0x0028

Bits	Field Name	Default NVM Value	Description
15:0	PCI_VFSUP H		

6.3.71.42 Address Low at TPH_CTRL - 0x0029

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of TPH_CTRL	0x11074	
3:0	Type	0x1	

6.3.71.43 Address High at TPH_CTRL - 0x002A

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of TPH_CTRL		

6.3.71.44 TPH_CTRL L - 0x002B

TPH Control Register

Bits	Field Name	Default NVM Value	Description
15:0	TPH_CTRL L	0x00001800	

6.3.71.45 TPH_CTRL H - 0x002C

Bits	Field Name	Default NVM Value	Description
15:0	TPH_CTRL H		

6.3.71.46 Address Low at PCI_LINKCAP - 0x002D

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_LINKCAP	0x11090	
3:0	Type	0x1	

6.3.71.47 Address High at PCI_LINKCAP - 0x002E

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_LINKCAP		

6.3.71.48 PCI_LINKCAP L - 0x002F

PCIe Link Capabilities

Bits	Field Name	Default NVM Value	Description
15:0	PCI_LINKCAP L	0x00000E80	

6.3.71.49 PCI_LINKCAP H - 0x0030

Bits	Field Name	Default NVM Value	Description
15:0	PCI_LINKCAP H		

6.3.71.50 Address Low at PCI_GLBL_CNF - 0x0031

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_GLBL_CNF	0x110B8	
3:0	Type	0x1	

6.3.71.51 Address High at PCI_GLBL_CNF - 0x0032

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_GLBL_CNF		

6.3.71.52 PCI_GLBL_CNF L - 0x0033

Wake# enable

Bits	Field Name	Default NVM Value	Description
15:8	PCIE_CLKGATE_TIMER	0x10	Clock gating idle timer. This field defines the number of clocks (CSR clock) of idle-detect before gating the PCIE clocks.
7:6	RESERVED	0x0	
5	PCIE_PCLK_GATE_EN	0x0	When set to 1b, and if PCIE_CLKGATE_DIS is 0, the PCIE PCLK will also be dynamically gated.
4	PCIE_CLKGATE_L1_ONLY	0x1	When set to 1b, and if PCIE_CLKGATE_DIS is 0, the clock gating of the PCIE clocks will be only when the PCIE is in L1 state.
3	PCIE_CLKGAT_DIS	0x1	When set to 0b enables dynamic clock gating of the PCIE clocks (LCB, HIU & CSR).
2	WAKE_PIN_EN	0x0	When set to 1b enables the use of PE_WAKE_N pin for PME event in all power states, otherwise, it is enabled only in Dr state.
1:0	RESERVED	0x0	

6.3.71.53 PCI_GLBL_CNF H - 0x0034

Bits	Field Name	Default NVM Value	Description
15:3	RESERVED	0x0	
2	DIS_SCREAM_ON_D ATA_PARITY	0x1	
1	DIS_SCREAM_ON_C MD_PARITY	0x1	
0	DIS_STOP_ON_CMD _PARITY	0x1	

6.3.71.54 Address Low at PCI_VENDORID - 0x0035

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_VENDORID	0x1103C	
3:0	Type	0x1	

6.3.71.55 Address High at PCI_VENDORID - 0x0036

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_VENDORID		

6.3.71.56 PCI_VENDORID L - 0x0037

Wake# enable

Bits	Field Name	Default NVM Value	Description
15:0	PCI_VENDORID L	0x8086	

6.3.71.57 PCI_VENDORID H - 0x0038

Bits	Field Name	Default NVM Value	Description
15:0	PCI_VENDORID H	0x0	

6.3.71.58 Address Low at PCI_PFDEVID - 0x0039

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_PFDEVID	0x11008	
3:0	Type	0x1	

6.3.71.59 Address High at PCI_PFDEVID - 0x003A

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_PFDEVID		

6.3.71.60 PCI_PFDEVID Low (LAN) - 0x003B

Bits	Field Name	Default NVM Value	Description
15:0	PF_DEV_ID_LAN	0x57AC	If the Load Device ID in offset 0x7 of section PCIe General configuration is set, this word is loaded to the device ID of the LAN function.

6.3.71.61 PCI_PFDEVID High (SAN) - 0x003C

Bits	Field Name	Default NVM Value	Description
15:0	PF_DEV_ID_SAN	0x57AC	

6.3.71.62 Address Low at PCI_CNF - 0x003D

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_CNF	0x11000	
3:0	Type	0x1	

6.3.71.63 Address High at PCI_CNF - 0x003E

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_CNF		

6.3.71.64 PCI_CNF Low - 0x003F

Bits	Field Name	Default NVM Value	Description
15:7	Reserved	0x0	Reserved
6:5	INT_PIN	0x0	Controls the value advertised in the Interrupt Pin field of the PCI configuration header for this function. Values of 00b, 01b, 10b and 11b correspond to INTA#, INTB#, INTC# and INTD# respectively. The value advertised in the PCI configuration header is the value loaded from NVM + 1.
4	IO_BAR	0x0	I/O BAR support 0b - I/O BAR is not supported 1b - I/O BAR is supported

Bits	Field Name	Default NVM Value	Description
3	EXROM_DIS	0x0	Expansion ROM Disable 0b - The Expansion ROM BAR in the PCI configuration space is enabled 1b - The Expansion ROM BAR in the PCI configuration space is disabled
2:0	Reserved	0x0	Reserved

6.3.71.65 PCI_CNF High - 0x0040

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	Reserved

6.3.71.66 Address Low at PCI_VFDEVID - 0x0041

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_VFDEVID	0x11010	
3:0	Type	0x1	

6.3.71.67 Address High at PCI_VFDEVID - 0x0042

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_VFDEVID		

6.3.71.68 PCI_VFDEVID Low (LAN) - 0x0043

Bits	Field Name	Default NVM Value	Description
15:0	VF_DEV_ID_LAN	0x57AD	If the Load Device ID in offset 0x7 of section PCIe General configuration is set, this word is loaded to the device ID of the LAN function.

6.3.71.69 PCI_VFDEVID High (SAN) - 0x0044

Bits	Field Name	Default NVM Value	Description
15:0	VF_DEV_ID_SAN	0x57AD	

6.3.71.70 Address Low at PCI_CLASS - 0x0045

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_CLASS	0x11038	
3:0	Type	0x1	

6.3.71.71 Address High at PCI_CLASS - 0x0046

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_CLASS		

6.3.71.72 PCI_CLASS Low - 0x0047

Bits	Field Name	Default NVM Value	Description
15:1	Reserved	0x0	Reserved
0	STORAGE_CLASS	0x0	0b - The class code of this port is set to 0x020000 (LAN) 1b - The class code of this port is set to 0x010000 (SCSI)

6.3.71.73 PCI_CLASS High - 0x0048

Bits	Field Name	Default NVM Value	Description
15:1	Reserved	0x0	Reserved
0	STORAGE_CLASS	0x0	0b - The class code of this port is set to 0x020000 (LAN) 1b - The class code of this port is set to 0x010000 (SCSI)

6.3.71.74 Address Low at PCI_FUNC2 - 0x0049

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_FUNC2	0x11018	
3:0	Type	0x1	

6.3.71.75 Address High at PCI_FUNC2 - 0x004A

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_FUNC2		

6.3.71.76 PCI_FUNC2 Low - 0x004B

Bits	Field Name	Default NVM Value	Description
15:1	Reserved	0x0	Reserved
0	FW_FUNC_DIS	0x0	FW Function Disable - Set by the FW when a CLP string or BMC requires that a function is disabled 0b - Function is enabled 1b - Function is disabled

6.3.71.77 PCI_FUNC2 High - 0x004C

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	Reserved

6.3.71.78 Address Low at PCI_PFDEVID - 0x004D

For inner structure please See [section 6.3.71.58](#)

6.3.71.79 Address High at PCI_PFDEVID - 0x004E

For inner structure please See [section 6.3.71.59](#)

6.3.71.80 PCI_PFDEVID Low (LAN) - 0x004F

For inner structure please See [section 6.3.71.60](#)

6.3.71.81 PCI_PFDEVID High (SAN) - 0x0050

For inner structure please See [section 6.3.71.61](#)

6.3.71.82 Address Low at PCI_CNF - 0x0051

For inner structure please See [section 6.3.71.62](#)

6.3.71.83 Address High at PCI_CNF - 0x0052

For inner structure please See [section 6.3.71.63](#)

6.3.71.84 PCI_CNF Low - 0x0053

For inner structure please See [section 6.3.71.64](#)

6.3.71.85 PCI_CNF High - 0x0054

For inner structure please See [section 6.3.71.65](#)

6.3.71.86 Address Low at PCI_VFDEVID - 0x0055

For inner structure please See [section 6.3.71.66](#)

6.3.71.87 Address High at PCI_VFDEVID - 0x0056

For inner structure please See [section 6.3.71.67](#)

6.3.71.88 PCI_VFDEVID Low (LAN) - 0x0057

For inner structure please See [section 6.3.71.68](#)

6.3.71.89 PCI_VFDEVID High (SAN) - 0x0058

For inner structure please See [section 6.3.71.69](#)

6.3.71.90 Address Low at PCI_CLASS - 0x0059

For inner structure please See [section 6.3.71.70](#)

6.3.71.91 Address High at PCI_CLASS - 0x005A

For inner structure please See [section 6.3.71.71](#)

6.3.71.92 PCI_CLASS Low - 0x005B

For inner structure please See [section 6.3.71.72](#)

6.3.71.93 PCI_CLASS High - 0x005C

For inner structure please See [section 6.3.71.73](#)

6.3.71.94 Address Low at PCI_FUNC2 - 0x005D

For inner structure please See [section 6.3.71.74](#)

6.3.71.95 Address High at PCI_FUNC2 - 0x005E

For inner structure please See [section 6.3.71.75](#)

6.3.71.96 PCI_FUNC2 Low - 0x005F

For inner structure please See [section 6.3.71.76](#)

6.3.71.97 PCI_FUNC2 High - 0x0060

For inner structure please See [section 6.3.71.77](#)

6.3.72 POR_PFA Autoload Allowlist Module Section Summary Table

Default setup to registers and internal memories that load on PCIR events

Word Offset	Description	Reference
0x0000	Module Length	See section 6.3.72.1
0x0001	Address Low at GL_MNG_HWARB_CTRL	See section 6.3.72.2
0x0002	Address High at GL_MNG_HWARB_CTRL	See section 6.3.72.3
0x0003	Data Low of GL_MNG_HWARB_CTRL	See section 6.3.72.4
0x0004	Data High of GL_MNG_HWARB_CTRL	See section 6.3.72.5
0x0005	Address Low at Port 0 PRTGEN_CNF	See section 6.3.72.6
0x0006	Address High at Port 0 PRTGEN_CNF	See section 6.3.72.7
0x0007	PRTGEN_CNF L	See section 6.3.72.8
0x0008	PRTGEN_CNF H	See section 6.3.72.9
0x0009	Address Low at Port 1 PRTGEN_CNF	See section 6.3.72.10
0x000A	Address High at Port 1 PRTGEN_CNF	See section 6.3.72.11
0x000B	PRTGEN_CNF L	See section 6.3.72.12
0x000C	PRTGEN_CNF H	See section 6.3.72.13
0x000D	Address Low at PF 0 PCI_FUNC	See section 6.3.72.14
0x000E	Address High at PF 0 PCI_FUNC	See section 6.3.72.15
0x000F	PCI_FUNC L	See section 6.3.72.16
0x0010	PCI_FUNC H	See section 6.3.72.17
0x0011	Address Low at PF 1 PCI_FUNC	See section 6.3.72.18
0x0012	Address High at PF 1 PCI_FUNC	See section 6.3.72.19
0x0013	PCI_FUNC L	See section 6.3.72.20
0x0014	PCI_FUNC H	See section 6.3.72.21
0x0015	Address Low at IOSFSB_LTR	See section 6.3.72.22
0x0016	Address High at IOSFSB_LTR	See section 6.3.72.23
0x0017	IOSFSB_LTR L	See section 6.3.72.24
0x0018	IOSFSB_LTR H	See section 6.3.72.25

6.3.72.1 Module Length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 1 First Section -> Word: POR_PFA Autoload Allowlist Module -> Module Length Last Section -> Word: POR_PFA Autoload Allowlist Module -> IOSFSB_LTR H

6.3.72.2 Address Low at GL_MNG_HWARB_CTRL - 0x0001

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of GL_MNG_HWARB_CTRL	0xB6130	
3:0	Type	0x1	

6.3.72.3 Address High at GL_MNG_HWARB_CTRL - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of GL_MNG_HWARB_CTRL		

6.3.72.4 Data Low of GL_MNG_HWARB_CTRL - 0x0003

Bits	Field Name	Default NVM Value	Description
15:1	RESERVED	0x0	
0	NCSI_ARB_EN	0x1	Hardware Arbitration Enable. If this bit is set, it is assumed the NCSI_ARB_IN and NCSI_ARB_OUT are connected to an Hardware arbitration ring. Otherwise, the NCSI_ARB_IN pin is pulled up internally.

6.3.72.5 Data High of GL_MNG_HWARB_CTRL - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	RESERVED cont		

6.3.72.6 Address Low at Port 0 PRTGEN_CNF - 0x0005

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTGEN_CNF	0x83120	
3:0	Type	0x1	

6.3.72.7 Address High at Port 0 PRTGEN_CNF - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTGEN_CNF		

6.3.72.8 PRTGEN_CNF L - 0x0007

Bits	Field Name	Default NVM Value	Description
15:3	RSVD	0x0	Reserved
2	EMP_PORT_DIS	0x0	Set by EMP to disable the Ethernet port. The NVM value for this bit should be 0b (enabled). NVM should use the PORT_DIS bit to disable a port
1	ALLOW_PORT_DIS	0x0	Allow Port Disable 0b = Asserting DEV_DIS_N has no effect on this port 1b = Asserting DEV_DIS_N disables this port
0	PORT_DIS	0x0	Port Disable. Defines if the Ethernet port is enabled from the NVM Exception: Port 0 is always enabled and cannot be disabled from the NVM 0b = Enabled 1b = Disabled

6.3.72.9 PRTGEN_CNF H - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	

6.3.72.10 Address Low at Port 1 PRTGEN_CNF - 0x0009

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PRTGEN_CNF	0x883120	
3:0	Type	0x1	

6.3.72.11 Address High at Port 1 PRTGEN_CNF - 0x000A

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PRTGEN_CNF		

6.3.72.12 PRTGEN_CNF L - 0x000B

Bits	Field Name	Default NVM Value	Description
15:3	RSVD	0x0	Reserved
2	EMP_PORT_DIS	0x0	Set by EMP to disable the Ethernet port. The NVM value for this bit should be 0b (enabled). NVM should use the PORT_DIS bit to disable a port
1	ALLOW_PORT_DIS	0x0	Allow Port Disable 0b = Asserting DEV_DIS_N has no effect on this port 1b = Asserting DEV_DIS_N disables this port

Bits	Field Name	Default NVM Value	Description
0	PORT_DIS	0x0	Port Disable. Defines if the Ethernet port is enabled from the NVM Exception: Port 0 is always enabled and cannot be disabled from the NVM 0b = Enabled 1b = Disabled

6.3.72.13 PRTGEN_CNF H - 0x000C

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	

6.3.72.14 Address Low at PF 0 PCI_FUNC - 0x000D

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of PCI_FUNC	0x011020	
3:0	Type	0x1	

6.3.72.15 Address High at PF 0 PCI_FUNC - 0x000E

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of PCI_FUNC		

6.3.72.16 PCI_FUNC L - 0x000F

Bits	Field Name	Default NVM Value	Description
15:3	RSVD	0x0	Reserved
2	DIS_FUNC_ON_PORT_DIS	0x0	Defines whether this PF is disabled when the DEV_DIS_N pin is asserted. 0b = Asserting DEV_DIS_N has no effect on this PCI function. 1b = Asserting DEV_DIS_N disables this PCI function.
1	ALLOW_FUNC_DIS	0x0	Allow Function Disable. 0b = Asserting PCI_DIS_N has no effect on this PCI function. 1b = Asserting PCI_DIS_N disables this PCI function.
0	FUNC_DIS	0x0	Function Disable. Defines if the PCI function is enabled from the NVM. Exception: This bit is RO for PF0. It is always enabled and cannot be disabled from the NVM. 0b = Enabled. 1b = Disabled. Default: 0b for PF0. 1b for other functions.

6.3.72.17 PCI_FUNC H - 0x0010

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	

6.3.72.18 Address Low at PF 1 PCI_FUNC - 0x0011

For inner structure please See [section 6.3.72.14](#)

6.3.72.19 Address High at PF 1 PCI_FUNC - 0x0012

For inner structure please See [section 6.3.72.15](#)

6.3.72.20 PCI_FUNC L - 0x0013

For inner structure please See [section 6.3.72.16](#)

6.3.72.21 PCI_FUNC H - 0x0014

For inner structure please See [section 6.3.72.17](#)

6.3.72.22 Address Low at IOSFSB_LTR - 0x0015

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of IOSFSB_LTR	0xA4240	
3:0	Type	0x1	

6.3.72.23 Address High at IOSFSB_LTR - 0x0016

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of IOSFSB_LTR		

6.3.72.24 IOSFSB_LTR L - 0x0017

Sideband IOSF LTR Control

Bits	Field Name	Default NVM Value	Description
15:0	IOSFSB_LTR L	0x0040	

6.3.72.25 IOSFSB_LTR H - 0x0018

Bits	Field Name	Default NVM Value	Description
15:0	IOSFSB_LTR H	0x0000	

6.3.73 MAC Reset Registers Autoload Module Section Summary Table

Default setup to registers and internal memories that load on MAC Reset events

Word Offset	Description	Reference
0x0000	Module Length	See section 6.3.73.1
0x0001	Starting Address Low at MACC	See section 6.3.73.2
0x0002	Starting Address High at MACC	See section 6.3.73.3
0x0003	Data low of MACC	See section 6.3.73.4
0x0004	Data high of MACC	See section 6.3.73.5
0x0005	Starting Address Low at MFLCN	See section 6.3.73.6
0x0006	Starting Address High at MFLCN	See section 6.3.73.7
0x0007	Data low of MFLCN	See section 6.3.73.8
0x0008	Data high of MFLCN	See section 6.3.73.9
0x0009	Starting Address Low at MACC	See section 6.3.73.10
0x000A	Starting Address High at MACC	See section 6.3.73.11
0x000B	Data low of MACC	See section 6.3.73.12
0x000C	Data high of MACC	See section 6.3.73.13
0x000D	Starting Address Low at MFLCN	See section 6.3.73.14
0x000E	Starting Address High at MFLCN	See section 6.3.73.15
0x000F	Data low of MFLCN	See section 6.3.73.16
0x0010	Data high of MFLCN	See section 6.3.73.17

6.3.73.1 Module Length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 1 First Section -> Word: MAC Reset Registers Autoload Module -> Module Length Last Section -> Word: MAC Reset Registers Autoload Module -> Data high of MFLCN

6.3.73.2 Starting Address Low at MACC - 0x0001

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of MACC	0x4330	
3:0	Type	0x1	

6.3.73.3 Starting Address High at MACC - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of MACC		

6.3.73.4 Data low of MACC - 0x0003

Bits	Field Name	Default NVM Value	Description
15:12	FIFOLT	0x3	FIFO Low Threshold. Determines the low threshold of the MAC elastic FIFO. Mapped to MACC.FIFOHT.
11:8	FIFOHT	0xD	FIFO High Threshold. Determines the high threshold of the MAC elastic FIFO. Mapped to MACC.FIFOLT.
7	Swap Rx Control	0x0	Swap the MAC internal Rx XGMII controls. 0b = Swap disabled, normal mode. 1b = Swap enabled. Mapped to MACC.Swap Rx Control. Valid values are: 0x0 = Swap disabled, normal mode. 0x1 = Swap enabled
6	Swap Rx Data	0x0	Swap the MAC internal Rx XGMII lanes. 0b = Swap disabled, normal mode. 1b = Swap enabled. Mapped to MACC.Swap Rx Data Valid values are: 0x0 = Swap disabled, normal mode. 0x1 = Swap enabled
5	Swizzle Rx Data	0x0	Swizzle the bytes in all 4 MAC internal Rx XGMII lanes. 0b = Swizzle disabled, normal mode. 1b = Swizzle enabled. Mapped to MACC.Swizzle Rx Data Valid values are: 0x0 = Swizzle disabled, normal mode. 0x1 = Swizzle enabled
4	Swap Tx Control	0x0	Swap the MAC internal Tx XGMII controls. 0b = Swap disabled, normal mode. 1b = Swap enabled. Mapped to MACC.Swap Tx Control Valid values are: 0x0 = Swap disabled, normal mode. 0x1 = Swap enabled
3	Swap Tx Data	0x0	Swap the MAC internal Tx XGMII lanes. 0b = Swap disabled, normal mode. 1b = Swap enabled. Mapped to MACC.Swap Tx Data Valid values are: 0x0 = Swap disabled, normal mode. 0x1 = Swap enabled
2	Swizzle Tx Data	0x0	Swizzle the bytes in all 4 MAC internal Tx XGMII lanes. 0b = Swizzle disabled, normal mode. 1b = Swizzle enabled. Mapped to MACC.Swizzle Tx Data Valid values are: 0x0 = Swizzle disabled, normal mode. 0x1 = Swizzle enabled
1	MAC_RX2TX_LPBK_EN	0x0	
0	FLU	0x0	

6.3.73.5 Data high of MACC - 0x0004

Bits	Field Name	Default NVM Value	Description
15:6	Reserved	0x0	Reserved
5:4	BLKXGMII	0x0	Block XGMII interface. 00 - Normal operation. The XGMII interface fed from the PCS while link is down. 01 - Idle state. The XGMII interface fed with Idles while link is down. 10 - Local Fault state. The XGMII interface fed with Local Fault sequences while link is down. 11 - Reserved Valid values are: 0x0 = Normal operation 0x1 = Idle state 0x2 = Local Fault state 0x3 = Reserved
3:0	Reserved	0x0	Reserved

6.3.73.6 Starting Address Low at MFLCN - 0x0005

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of MFLCN	0x4294	
3:0	Type	0x1	

6.3.73.7 Starting Address High at MFLCN - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of MFLCN		

6.3.73.8 Data low of MFLCN - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	Data low of MFLCN	0x2	

6.3.73.9 Data high of MFLCN - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	Data high of MFLCN	0x0	

6.3.73.10 Starting Address Low at MACC - 0x0009

For inner structure please See [section 6.3.73.2](#)

6.3.73.11 Starting Address High at MACC - 0x000A

For inner structure please See [section 6.3.73.3](#)

6.3.73.12 Data low of MACC - 0x000B

For inner structure please See [section 6.3.73.4](#)

6.3.73.13 Data high of MACC - 0x000C

For inner structure please See [section 6.3.73.5](#)

6.3.73.14 Starting Address Low at MFLCN - 0x000D

For inner structure please See [section 6.3.73.6](#)

6.3.73.15 Starting Address High at MFLCN - 0x000E

For inner structure please See [section 6.3.73.7](#)

6.3.73.16 Data low of MFLCN - 0x000F

For inner structure please See [section 6.3.73.8](#)

6.3.73.17 Data high of MFLCN - 0x0010

For inner structure please See [section 6.3.73.9](#)

6.3.74 Master Reset Registers Autoload Module Section Summary Table

Default setup to registers and internal memories that load on Master Reset events

Word Offset	Description	Reference
0x0000	Module Length	See section 6.3.74.1
0x0001	Address Low at LEDCTL	See section 6.3.74.2
0x0002	Address High at LEDCTL	See section 6.3.74.3
0x0003	LED Control Lower Word	See section 6.3.74.4
0x0004	LED Control Upper Word	See section 6.3.74.5
0x0005	Address Low at ESDP	See section 6.3.74.6
0x0006	Address High at ESDP	See section 6.3.74.7
0x0007	SDP Control	See section 6.3.74.8
0x0008	Reserved	See section 6.3.74.9
0x0009	Address Low at WUC	See section 6.3.74.10
0x000A	Address High at WUC	See section 6.3.74.11
0x000B	Filter Control	See section 6.3.74.12
0x000C	Reserved	See section 6.3.74.13
0x000D	Address Low at LEDCTL	See section 6.3.74.14
0x000E	Address High at LEDCTL	See section 6.3.74.15
0x000F	LED Control Lower Word	See section 6.3.74.16
0x0010	LED Control Upper Word	See section 6.3.74.17
0x0011	Address Low at ESDP	See section 6.3.74.18
0x0012	Address High at ESDP	See section 6.3.74.19
0x0013	SDP Control	See section 6.3.74.20
0x0014	Reserved	See section 6.3.74.21
0x0015	Address Low at WUC	See section 6.3.74.22
0x0016	Address High at WUC	See section 6.3.74.23
0x0017	Filter Control	See section 6.3.74.24
0x0018	Reserved	See section 6.3.74.25

6.3.74.1 Module Length - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Module Length		Length in: 2 Bytes unit - 1 First Section -> Word: MAC Reset Registers Autoload Module -> Module Length Last Section -> Word: MAC Reset Registers Autoload Module -> Data high of MFLCN

6.3.74.2 Address Low at LEDCTL - 0x0001

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of LEDCTL	0x0200	
3:0	Type	0x1	

6.3.74.3 Address High at LEDCTL - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of LEDCTL		

6.3.74.4 LED Control Lower Word - 0x0003

The LEDCTL register defaults are loaded from two words as listed in the following tables.

Bits	Field Name	Default NVM Value	Description
15	LED1_BLINK	0x1	Valid values are: 0x0 = Don't Blink 0x1 = Blink
14	LED1_IVRT	0x0	Valid values are: 0x0 = Active Low 0x1 = Active High
13:12	Reserved	0x0	
11:8	LED1_MODE	0x1	LED 0 Control Valid values are: 0x0 = LINK_UP 0x1 = LINK 10G 0x2 = MAC ACTIVITY 0x3 = FILTER ACTIVITY 0x4 = LINK/ACTIVITY 0x5 = LINK 1G 0x6 = LINK 100M 0xE = LED ON 0xF = LED OFF
7	LED0_BLINK	0x0	Valid values are: 0x0 = Don't Blink 0x1 = Blink
6	LED0_IVRT	0x0	Valid values are: 0x0 = Active Low 0x1 = Active High
5	Global Link Mode	0x0	Valid values are: 0x0 = 200 ms 0x1 = 83 ms
4	Reserved	0x0	

Bits	Field Name	Default NVM Value	Description
3:0	LED0_MODE	0x0	LED 0 Control Valid values are: 0x0 = LINK_UP 0x1 = LINK 10G 0x2 = MAC ACTIVITY 0x3 = FILTER ACTIVITY 0x4 = LINK/ACTIVITY 0x5 = LINK 1G 0x6 = LINK 100M 0xE = LED ON 0xF = LED OFF

6.3.74.5 LED Control Upper Word - 0x0004

Bits	Field Name	Default NVM Value	Description
15	LED3_BLINK	0x0	Valid values are: 0x0 = Don't Blink 0x1 = Blink
14	LED3_IVRT	0x0	Valid values are: 0x0 = Active Low 0x1 = Active High
13:12	Reserved	0x0	
11:8	LED3_MODE	0x0	LED 0 Control Valid values are: 0x0 = LINK_UP 0x1 = LINK 10G 0x2 = MAC ACTIVITY 0x3 = FILTER ACTIVITY 0x4 = LINK/ACTIVITY 0x5 = LINK 1G 0x6 = LINK 100M 0xE = LED ON 0xF = LED OFF
7	LED2_BLINK	0x0	Valid values are: 0x0 = Don't Blink 0x1 = Blink
6	LED2_IVRT	0x0	Valid values are: 0x0 = Active Low 0x1 = Active High
5:4	Reserved	0x0	
3:0	LED2_MODE	0x0	LED 0 Control Valid values are: 0x0 = LINK_UP 0x1 = LINK 10G 0x2 = MAC ACTIVITY 0x3 = FILTER ACTIVITY 0x4 = LINK/ACTIVITY 0x5 = LINK 1G 0x6 = LINK 100M 0xE = LED ON 0xF = LED OFF

6.3.74.6 Address Low at ESDP - 0x0005

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of ESDP	0x0020	
3:0	Type	0x1	

6.3.74.7 Address High at ESDP - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of ESDP		

6.3.74.8 SDP Control - 0x0007

Bits	Field Name	Default NVM Value	Description
15	SDP3_NATIVE	0x0	Defines SDP1 operating mode is mapped to ESDP.SDP1_NATIVE loaded at power up: 0 - Operates as generic software controlled IO. 1 - Native mode operation (hardware function). Valid values are: 0x0 = Generic software controlled IO 0x1 = Native mode operation (hardware function)
14	SDP2_NATIVE	0x0	Defines SDP1 operating mode is mapped to ESDP.SDP1_NATIVE loaded at power up: 0 - Operates as generic software controlled IO. 1 - Native mode operation (hardware function). Valid values are: 0x0 = Generic software controlled IO 0x1 = Native mode operation (hardware function)
13	SDP1_NATIVE	0x0	Defines SDP1 operating mode is mapped to ESDP.SDP1_NATIVE loaded at power up: 0 - Operates as generic software controlled IO. 1 - Native mode operation (hardware function). Valid values are: 0x0 = Generic software controlled IO 0x1 = Native mode operation (hardware function)
12	SDP0_NATIVE	0x0	Defines SDP1 operating mode is mapped to ESDP.SDP1_NATIVE loaded at power up: 0 - Operates as generic software controlled IO. 1 - Native mode operation (hardware function). Valid values are: 0x0 = Generic software controlled IO 0x1 = Native mode operation (hardware function)
11	SDPDIR[3]	0x0	SDP3 Pin - Initial Direction is mapped to ESDP.SDP3_IODIR loaded at power up.
10	SDPDIR[2]	0x0	SDP2 Pin - Initial Direction is mapped to ESDP.SDP2_IODIR loaded at power up.
9	SDPDIR[1]	0x0	SDP1 Pin - Initial Direction is mapped to ESDP.SDP1_IODIR loaded at power up.
8	SDPDIR[0]	0x0	SDP0 Pin - Initial Direction is mapped to ESDP.SDP0_IODIR loaded at power up.
7	Reserved	0x0	Reserved
6	REVERSE_DIR	0x0	When set reverses the polarity of the pin direction signal to pads block, such that pads are fed with output enable signal instead of output disable. Should be cleared for normal operation.

Bits	Field Name	Default NVM Value	Description
5	SDP23_function	0x0	Defines the initial value for ESDP.SDP23_function 0 = 1588 functionality 1 = i2c functionality. Relevant only if SDP2_NATIVE or SDP3_NATIVE is set. If this bit is set, then SDP2_NATIVE and SDP3_NATIVE should have the same value.
4	SDP1_function	0x0	Defines the initial value for ESDP.SDP1_function 0b = Time Sync functionality. 1b = SDP1 Thermal sensor functionality. SDPDIR[1] should be configured as an output. Relevant only if SDP1_NATIVE is set.
3	SDPVAL[3]	0x0	SDP3 Pin - Initial Output Value is mapped to ESDP.SDP3_DATA loaded at power up.
2	SDPVAL[2]	0x0	SDP2 Pin - Initial Output Value is mapped to ESDP.SDP2_DATA loaded at power up.
1	SDPVAL[1]	0x0	SDP1 Pin - Initial Output Value is mapped to ESDP.SDP1_DATA loaded at power up.
0	SDPVAL[0]	0x0	SDP0 Pin - Initial Output Value is mapped to ESDP.SDP0_DATA loaded at power up.

6.3.74.9 Reserved - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	

6.3.74.10 Address Low at WUC - 0x0009

Bits	Field Name	Default NVM Value	Description
15:4	Low Address Bits of WUC	0x5800	
3:0	Type	0x1	

6.3.74.11 Address High at WUC - 0x000A

Bits	Field Name	Default NVM Value	Description
15:0	High Address Bits of WUC		

6.3.74.12 Filter Control - 0x000B

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0001	Reserved

6.3.74.13 Reserved - 0x000C

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0x0	

6.3.74.14 Address Low at LEDCTL - 0x000D

For inner structure please See [section 6.3.74.2](#)

6.3.74.15 Address High at LEDCTL - 0x000E

For inner structure please See [section 6.3.74.3](#)

6.3.74.16 LED Control Lower Word - 0x000F

The LEDCTL register defaults are loaded from two words as listed in the following tables.

For inner structure please See [section 6.3.74.4](#)

6.3.74.17 LED Control Upper Word - 0x0010

For inner structure please See [section 6.3.74.5](#)

6.3.74.18 Address Low at ESDP - 0x0011

For inner structure please See [section 6.3.74.6](#)

6.3.74.19 Address High at ESDP - 0x0012

For inner structure please See [section 6.3.74.7](#)

6.3.74.20 SDP Control - 0x0013

For inner structure please See [section 6.3.74.8](#)

6.3.74.21 Reserved - 0x0014

For inner structure please See [section 6.3.74.9](#)

6.3.74.22 Address Low at WUC - 0x0015

For inner structure please See [section 6.3.74.10](#)

6.3.74.23 Address High at WUC - 0x0016

For inner structure please See [section 6.3.74.11](#)

6.3.74.24 Filter Control - 0x0017

For inner structure please See [section 6.3.74.12](#)

6.3.74.25 Reserved - 0x0018

For inner structure please See [section 6.3.74.13](#)

6.3.75 NVM ISK Object CSS Header Section Summary Table

Word Offset	Description	Reference
0x0000	moduleTypeL	See section 6.3.75.1
0x0001	moduleTypeH	See section 6.3.75.2
0x0002	headerLenL	See section 6.3.75.3
0x0003	headerLenH	See section 6.3.75.4
0x0004	headerVersionL	See section 6.3.75.5
0x0005	headerVersionH	See section 6.3.75.6
0x0006	moduleIDL	See section 6.3.75.7
0x0007	moduleIDH	See section 6.3.75.8
0x0008	moduleVendorL	See section 6.3.75.9
0x0009	moduleVendorH	See section 6.3.75.10
0x000A	dateL	See section 6.3.75.11
0x000B	dateH	See section 6.3.75.12
0x000C	sizeL	See section 6.3.75.13
0x000D	sizeH	See section 6.3.75.14
0x000E	keySizeL	See section 6.3.75.15
0x000F	keySizeH	See section 6.3.75.16
0x0010	modulusSizeL	See section 6.3.75.17
0x0011	modulusSizeH	See section 6.3.75.18
0x0012	exponentSizeL	See section 6.3.75.19
0x0013	exponentSizeH	See section 6.3.75.20
0x0014	lad_srevL	See section 6.3.75.21
0x0015	lad_srevH	See section 6.3.75.22
0x0016	crypto_infoL	See section 6.3.75.23
0x0017	crypto_infoH	See section 6.3.75.24
0x0018	lad_fw_entry_offsetL	See section 6.3.75.25
0x0019	lad_fw_entry_offsetH	See section 6.3.75.26
0x001A	reserved2L	See section 6.3.75.27
0x001B	reserved2H	See section 6.3.75.28
0x001C	lad_image_unique_idL	See section 6.3.75.29
0x001D	lad_image_unique_idH	See section 6.3.75.30
0x001E	lad_module_idL	See section 6.3.75.31
0x001F	lad_module_idH	See section 6.3.75.32
0x0020 + 1*n, n=0...31	reserved	See section 6.3.75.33

6.3.75.1 moduleTypeL - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	moduleType	0x6	

6.3.75.2 moduleTypeH - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	moduleTypeH		

6.3.75.3 headerLenL - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	headerLenL	0x1A1	

6.3.75.4 headerLenH - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	headerLenH		

6.3.75.5 headerVersionL - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionL	0x00010000	

6.3.75.6 headerVersionH - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionH		

6.3.75.7 moduleIDL - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	moduleIDL	0x00	

6.3.75.8 moduleIDH - 0x0007

Bits	Field Name	Default NVM Value	Description
15	signMode	0x1	
14:0	moduleIDH		

6.3.75.9 moduleVendorL - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorL	0x00008086	

6.3.75.10 moduleVendorH - 0x0009

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorH		

6.3.75.11 dateL - 0x000A

0xMMDD

Bits	Field Name	Default NVM Value	Description
15:0	DateL	0x20130530	0xMMDD

6.3.75.12 dateH - 0x000B

0xYYYY

Bits	Field Name	Default NVM Value	Description
15:0	DateH		

6.3.75.13 sizeL - 0x000C

Bits	Field Name	Default NVM Value	Description
15:0	sizeL	0x1B6	

6.3.75.14 sizeH - 0x000D

Bits	Field Name	Default NVM Value	Description
15:0	sizeH		

6.3.75.15 keySizeL - 0x000E

Bits	Field Name	Default NVM Value	Description
15:0	keySizeL	0x60	

6.3.75.16 keySizeH - 0x000F

Bits	Field Name	Default NVM Value	Description
15:0	keySizeH		

6.3.75.17 modulusSizeL - 0x0010

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeL	0x60	

6.3.75.18 modulusSizeH - 0x0011

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeH		

6.3.75.19 exponentSizeL - 0x0012

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeL	0x1	

6.3.75.20 exponentSizeH - 0x0013

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeH		

6.3.75.21 lad_srevL - 0x0014

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevL	0x0	

6.3.75.22 lad_srevH - 0x0015

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevH		

6.3.75.23 crypto_infoL - 0x0016

Bits	Field Name	Default NVM Value	Description
15:8	hash algorithm	0x1	
7:0	sign mode	0x8	

6.3.75.24 crypto_infoH - 0x0017

Bits	Field Name	Default NVM Value	Description
15:8	reserved	0x0	
7:0	encryption algorithm	0x0	

6.3.75.25 lad_fw_entry_offsetL - 0x0018

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetL	0x34A	

6.3.75.26 lad_fw_entry_offsetH - 0x0019

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetH		

6.3.75.27 reserved2L - 0x001A

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.75.28 reserved2H - 0x001B

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.75.29 lad_image_unique_idL - 0x001C

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idL	0x0	

6.3.75.30 lad_image_unique_idH - 0x001D

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idH		

6.3.75.31 lad_module_idL - 0x001E

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idL	0xF0	Valid values are: 0x1 = EMP Image 0x2 = PE Image 0x3 = PCIe Analog 0x4 = PHY Analog 0x5 = Option ROM 0xB = Recovery 0xF0 = ISK Object

6.3.75.32 lad_module_idH - 0x001F

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idH		

6.3.75.33 reserved[n] (0x0020 + 1*n, n=0...31)

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.76 NVM ISK Object Key and Signature Section Summary Table

Word Offset	Description	Reference
0x0000 + 1*n, n=0...191	RSA Public Key	See section 6.3.76.1
0x00C0 + 1*n, n=0...191	RSA Public Key Prime	See section 6.3.76.2
0x0180 + 1*n, n=0...191	RSA Public Key Second	See section 6.3.76.3
0x0240	RSA ExponentL	See section 6.3.76.4
0x0241	RSA ExponentH	See section 6.3.76.5
0x0242 + 1*n, n=0...191	Encrypted SHA384 Hash	See section 6.3.76.6

6.3.76.1 RSA Public Key[n] (0x0000 + 1*n, n=0...191)

Associated with the public portion of the Key Authentication Key used to validate this ISK.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.3.76.2 RSA Public Key Prime[n] (0x00C0 + 1*n, n=0...191)

Associated with the public portion of the Key Authentication Key used to validate this ISK.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key Prime		

6.3.76.3 RSA Public Key Second[n] (0x0180 + 1*n, n=0...191)

Associated with the public portion of the Key Authentication Key used to validate this ISK.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key Second		

6.3.76.4 RSA ExponentL - 0x0240

Associated with the public portion of the Key Authentication Key used to validate this ISK.

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentL	0x0	

6.3.76.5 RSA ExponentH - 0x0241

Associated with the public portion of the Key Authentication Key used to validate this ISK.

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentH	0x0	

6.3.76.6 Encrypted SHA384 Hash[n] (0x0242 + 1*n, n=0...191)

Bits	Field Name	Default NVM Value	Description
15:0	Encrypted SHA384 Hash	0x0	

6.3.77 NVM ISK Object Auth Header Section Summary Table

Word Offset	Description	Reference
0x0000	Device Blank NVM Device ID	See section 6.3.77.1
0x0001	Max Module AreaL	See section 6.3.77.2
0x0002	Max Module AreaH	See section 6.3.77.3
0x0003	Current Module AreaL	See section 6.3.77.4
0x0004	Current Module AreaH	See section 6.3.77.5
0x0005	Reserved	See section 6.3.77.6
0x0006	Code Revision	See section 6.3.77.7
0x0007	Reserved Spare Word	See section 6.3.77.8

6.3.77.1 Device Blank NVM Device ID - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Device Blank NVM Device ID	0x57AC	

6.3.77.2 Max Module AreaL - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaL	0x36C	

6.3.77.3 Max Module AreaH - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaH	0x0	

6.3.77.4 Current Module AreaL - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaL	0x36C	

6.3.77.5 Current Module AreaH - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaH	0x0000	

6.3.77.6 Reserved - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	

6.3.77.7 Code Revision - 0x0006

Bits	Field Name	Default NVM Value	Description
15:8	Major Revision	0x0	
7:0	Minor Revision	0x0	

6.3.77.8 Reserved Spare Word - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	Reserved Spare Word	0x0	

6.3.78 NVM ISK Object Section Summary Table

Word Offset	Description	Reference
0x0000	ISK ID	See section 6.3.78.1
0x0001 + 1*n, n=0...31	ISK SHA512 Hash	See section 6.3.78.2

6.3.78.1 ISK ID - 0x0000

Bits	Field Name	Default NVM Value	Description
15	Revocation Scheme	0x1	0x0 - Pseudo timer based revocation 0x1 - Explicit revocation
14:0	Key Revocation ID	0x0	Indicates the index of the fuse used to revoke this ISK.

6.3.78.2 ISK SHA512 Hash[n] (0x0001 + 1*n, n=0...31)

Bits	Field Name	Default NVM Value	Description
15:0	ISK SHA512 Hash	0x0	

6.3.79 Signature RoT Key Revocation Module CSS Header Section Summary Table

Word Offset	Description	Reference
0x0000	moduleTypeL	See section 6.3.79.1
0x0001	moduleTypeH	See section 6.3.79.2
0x0002	headerLenL	See section 6.3.79.3
0x0003	headerLenH	See section 6.3.79.4
0x0004	headerVersionL	See section 6.3.79.5
0x0005	headerVersionH	See section 6.3.79.6
0x0006	moduleIDL	See section 6.3.79.7
0x0007	moduleIDH	See section 6.3.79.8
0x0008	moduleVendorL	See section 6.3.79.9
0x0009	moduleVendorH	See section 6.3.79.10
0x000A	dateL	See section 6.3.79.11
0x000B	dateH	See section 6.3.79.12
0x000C	sizeL	See section 6.3.79.13
0x000D	sizeH	See section 6.3.79.14
0x000E	keySizeL	See section 6.3.79.15
0x000F	keySizeH	See section 6.3.79.16
0x0010	modulusSizeL	See section 6.3.79.17
0x0011	modulusSizeH	See section 6.3.79.18
0x0012	exponentSizeL	See section 6.3.79.19
0x0013	exponentSizeH	See section 6.3.79.20
0x0014	lad_srevL	See section 6.3.79.21
0x0015	lad_srevH	See section 6.3.79.22
0x0016	crypto_infoL	See section 6.3.79.23
0x0017	crypto_infoH	See section 6.3.79.24
0x0018	lad_fw_entry_offsetL	See section 6.3.79.25
0x0019	lad_fw_entry_offsetH	See section 6.3.79.26
0x001A	reserved2L	See section 6.3.79.27
0x001B	reserved2H	See section 6.3.79.28
0x001C	lad_image_unique_idL	See section 6.3.79.29
0x001D	lad_image_unique_idH	See section 6.3.79.30
0x001E	lad_module_idL	See section 6.3.79.31
0x001F	lad_module_idH	See section 6.3.79.32
0x0020 + 1*n, n=0...31	reserved	See section 6.3.79.33

6.3.79.1 moduleTypeL - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	moduleType	0x6	

6.3.79.2 moduleTypeH - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	moduleTypeH		

6.3.79.3 headerLenL - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	headerLenL	0x1A1	

6.3.79.4 headerLenH - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	headerLenH		

6.3.79.5 headerVersionL - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionL	0x00010000	

6.3.79.6 headerVersionH - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionH		

6.3.79.7 moduleIDL - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	moduleIDL	0x00	

6.3.79.8 moduleIDH - 0x0007

Bits	Field Name	Default NVM Value	Description
15	signMode	0x1	
14:0	moduleIDH		

6.3.79.9 moduleVendorL - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorL	0x00008086	

6.3.79.10 moduleVendorH - 0x0009

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorH		

6.3.79.11 dateL - 0x000A

0xMMDD

Bits	Field Name	Default NVM Value	Description
15:0	DateL	0x20130530	0xMMDD

6.3.79.12 dateH - 0x000B

0xYYYY

Bits	Field Name	Default NVM Value	Description
15:0	DateH		

6.3.79.13 sizeL - 0x000C

Bits	Field Name	Default NVM Value	Description
15:0	sizeL	0x1AF	

6.3.79.14 sizeH - 0x000D

Bits	Field Name	Default NVM Value	Description
15:0	sizeH		

6.3.79.15 keySizeL - 0x000E

Bits	Field Name	Default NVM Value	Description
15:0	keySizeL	0x60	

6.3.79.16 keySizeH - 0x000F

Bits	Field Name	Default NVM Value	Description
15:0	keySizeH		

6.3.79.17 modulusSizeL - 0x0010

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeL	0x60	

6.3.79.18 modulusSizeH - 0x0011

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeH		

6.3.79.19 exponentSizeL - 0x0012

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeL	0x1	

6.3.79.20 exponentSizeH - 0x0013

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeH		

6.3.79.21 lad_srevL - 0x0014

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevL	0x0	

6.3.79.22 lad_srevH - 0x0015

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevH		

6.3.79.23 crypto_infoL - 0x0016

Bits	Field Name	Default NVM Value	Description
15:8	hash algorithm	0x1	
7:0	sign mode	0x8	

6.3.79.24 crypto_infoH - 0x0017

Bits	Field Name	Default NVM Value	Description
15:12	reserved	0x0	
11:8	srev_flags	0x0	
7:0	encryption algorithm	0x0	

6.3.79.25 lad_fw_entry_offsetL - 0x0018

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetL	0x34A	

6.3.79.26 lad_fw_entry_offsetH - 0x0019

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetH		

6.3.79.27 reserved2L - 0x001A

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.79.28 reserved2H - 0x001B

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.79.29 lad_image_unique_idL - 0x001C

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idL	0x0	

6.3.79.30 lad_image_unique_idH - 0x001D

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idH		

6.3.79.31 lad_module_idL - 0x001E

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idL	0xF2	Valid values are: 0x1 = EMP Image 0x2 = PE Image 0x3 = PCIe Analog 0x4 = PHY Analog 0x5 = Option ROM 0xB = Recovery 0xF0 = ISK Object 0xF2 = Key Revocation Module

6.3.79.32 lad_module_idH - 0x001F

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idH		

6.3.79.33 reserved[n] (0x0020 + 1*n, n=0...31)

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.80 Signature RoT Key Revocation Module Key and Signature Section Summary Table

Word Offset	Description	Reference
0x0000 + 1*n, n=0...191	RSA Public Key	See section 6.3.80.1
0x00C0 + 1*n, n=0...191	RSA Public Key Prime	See section 6.3.80.2
0x0180 + 1*n, n=0...191	RSA Public Key Second	See section 6.3.80.3
0x0240	RSA ExponentL	See section 6.3.80.4
0x0241	RSA ExponentH	See section 6.3.80.5
0x0242 + 1*n, n=0...191	Encrypted SHA384 Hash	See section 6.3.80.6

6.3.80.1 RSA Public Key[n] (0x0000 + 1*n, n=0...191)

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.3.80.2 RSA Public Key Prime[n] (0x00C0 + 1*n, n=0...191)

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key Prime	0x0	

6.3.80.3 RSA Public Key Second[n] (0x0180 + 1*n, n=0...191)

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key Second	0x0	

6.3.80.4 RSA ExponentL - 0x0240

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentL	0x0	

6.3.80.5 RSA ExponentH - 0x0241

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentH	0x0	

6.3.80.6 Encrypted SHA384 Hash[n] (0x0242 + 1*n, n=0...191)

Bits	Field Name	Default NVM Value	Description
15:0	Encrypted SHA384 Hash	0x0	

6.3.81 Signature RoT Key Revocation Module Auth Header Section Summary Table

Word Offset	Description	Reference
0x0000	Device Blank NVM Device ID	See section 6.3.81.1
0x0001	Max Module AreaL	See section 6.3.81.2
0x0002	Max Module AreaH	See section 6.3.81.3
0x0003	Current Module AreaL	See section 6.3.81.4
0x0004	Current Module AreaH	See section 6.3.81.5
0x0005	Reserved	See section 6.3.81.6
0x0006	Code Revision	See section 6.3.81.7
0x0007	Reserved Spare Word	See section 6.3.81.8

6.3.81.1 Device Blank NVM Device ID - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Device Blank NVM Device ID	0x57AC	

6.3.81.2 Max Module AreaL - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaL	0x35E	

6.3.81.3 Max Module AreaH - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaH	0x0	

6.3.81.4 Current Module AreaL - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaL	0x35E	

6.3.81.5 Current Module AreaH - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaH	0x0000	

6.3.81.6 Reserved - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	

6.3.81.7 Code Revision - 0x0006

Bits	Field Name	Default NVM Value	Description
15:8	Major Revision	0x0	
7:0	Minor Revision	0x0	

6.3.81.8 Reserved Spare Word - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	Reserved Spare Word	0x0	

6.3.82 Signature RoT Key Revocation Module Data Section Summary Table

Word Offset	Description	Reference
0x0000	Revoke KAK	See section 6.3.82.1
0x0001	Reserved	See section 6.3.82.2
0x0002	Reserved	See section 6.3.82.3
0x0003 + 1*n, n=0...15	Revoke ISK	See section 6.3.82.4

6.3.82.1 Revoke KAK - 0x0000

Bits	Field Name	Default NVM Value	Description
15:3	RESERVED	0x0	
2	Revoke KAK2	0x0	If set, image allows for revocation of KAK2.
1	Revoke KAK1	0x0	If set, image allows for the revocation of KAK1.
0	Revoke KAK0	0x0	If set, image allows for the revocation of KAK0.

6.3.82.2 Reserved - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	RESERVED	0x0	

6.3.82.3 Reserved - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	RESERVED	0x0	

6.3.82.4 Revoke ISK[n] (0x0003 + 1*n, n=0...15)

Bits	Field Name	Default NVM Value	Description
15:0	Revoke ISK	0x0	If set, image allows for revocation of corresponding ISK.

6.3.83 Recovery FW CSS Header Section Summary Table

Word Offset	Description	Reference
0x0000	moduleTypeL	See section 6.3.83.1
0x0001	moduleTypeH	See section 6.3.83.2
0x0002	headerLenL	See section 6.3.83.3
0x0003	headerLenH	See section 6.3.83.4
0x0004	headerVersionL	See section 6.3.83.5
0x0005	headerVersionH	See section 6.3.83.6
0x0006	moduleIDL	See section 6.3.83.7
0x0007	moduleIDH	See section 6.3.83.8
0x0008	moduleVendorL	See section 6.3.83.9
0x0009	moduleVendorH	See section 6.3.83.10
0x000A	dateL	See section 6.3.83.11
0x000B	dateH	See section 6.3.83.12
0x000C	sizeL	See section 6.3.83.13
0x000D	sizeH	See section 6.3.83.14
0x000E	keySizeL	See section 6.3.83.15
0x000F	keySizeH	See section 6.3.83.16
0x0010	modulusSizeL	See section 6.3.83.17
0x0011	modulusSizeH	See section 6.3.83.18
0x0012	exponentSizeL	See section 6.3.83.19
0x0013	exponentSizeH	See section 6.3.83.20
0x0014	lad_srevL	See section 6.3.83.21
0x0015	lad_srevH	See section 6.3.83.22
0x0016	crypto_infoL	See section 6.3.83.23
0x0017	crypto_infoH	See section 6.3.83.24
0x0018	lad_fw_entry_offsetL	See section 6.3.83.25
0x0019	lad_fw_entry_offsetH	See section 6.3.83.26
0x001A	reserved2L	See section 6.3.83.27
0x001B	reserved2H	See section 6.3.83.28
0x001C	lad_image_unique_idL	See section 6.3.83.29
0x001D	lad_image_unique_idH	See section 6.3.83.30
0x001E	lad_module_idL	See section 6.3.83.31
0x001F	lad_module_idH	See section 6.3.83.32
0x0020 + 1*n, n=0...31	reserved	See section 6.3.83.33

6.3.83.1 moduleTypeL - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	moduleType	0x6	

6.3.83.2 moduleTypeH - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	moduleTypeH		

6.3.83.3 headerLenL - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	headerLenL	0x1a1	

6.3.83.4 headerLenH - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	headerLenH		

6.3.83.5 headerVersionL - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionL	0x00010000	

6.3.83.6 headerVersionH - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionH		

6.3.83.7 moduleIDL - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	moduleIDL	0x00	

6.3.83.8 moduleIDH - 0x0007

Bits	Field Name	Default NVM Value	Description
15	signMode	0x1	
14:0	moduleIDH		

6.3.83.9 moduleVendorL - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorL	0x00008086	

6.3.83.10 moduleVendorH - 0x0009

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorH		

6.3.83.11 dateL - 0x000A

0xMMDD

Bits	Field Name	Default NVM Value	Description
15:0	DateL	0x20130530	0xMMDD

6.3.83.12 dateH - 0x000B

0xYYYY

Bits	Field Name	Default NVM Value	Description
15:0	DateH		

6.3.83.13 sizeL - 0x000C

Bits	Field Name	Default NVM Value	Description
15:0	sizeL	0x0000FA00	

6.3.83.14 sizeH - 0x000D

Bits	Field Name	Default NVM Value	Description
15:0	sizeH		

6.3.83.15 keySizeL - 0x000E

Bits	Field Name	Default NVM Value	Description
15:0	keySizeL	0x60	

6.3.83.16 keySizeH - 0x000F

Bits	Field Name	Default NVM Value	Description
15:0	keySizeH		

6.3.83.17 modulusSizeL - 0x0010

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeL	0x60	

6.3.83.18 modulusSizeH - 0x0011

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeH		

6.3.83.19 exponentSizeL - 0x0012

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeL	0x1	

6.3.83.20 exponentSizeH - 0x0013

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeH		

6.3.83.21 lad_srevL - 0x0014

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevL	0x1	

6.3.83.22 lad_srevH - 0x0015

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevH		

6.3.83.23 crypto_infoL - 0x0016

Bits	Field Name	Default NVM Value	Description
15:8	hash algorithm	0x1	
7:0	sign mode	0x8	

6.3.83.24 crypto_infoH - 0x0017

Bits	Field Name	Default NVM Value	Description
15:12	reserved	0x0	
11:8	srev_flags	0xC	
7:0	encryption algorithm	0x0	

6.3.83.25 lad_fw_entry_offsetL - 0x0018

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetL	0x6C0	

6.3.83.26 lad_fw_entry_offsetH - 0x0019

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetH		

6.3.83.27 reserved2L - 0x001A

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.83.28 reserved2H - 0x001B

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.83.29 lad_image_unique_idL - 0x001C

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idL	0x0	

6.3.83.30 lad_image_unique_idH - 0x001D

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idH		

6.3.83.31 lad_module_idL - 0x001E

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idL	0xB	Valid values are: 0x1 = EMP Image 0x2 = PE Image 0x3 = PCIe Analog 0x4 = PHY Analog 0x5 = Option ROM 0xB = Recovery

6.3.83.32 lad_module_idH - 0x001F

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idH		

6.3.83.33 reserved[n] (0x0020 + 1*n, n=0...31)

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.84 Recovery FW Key and Signature Section Summary Table

Word Offset	Description	Reference
0x0000 + 1*n, n=0...191	RSA Public Key	See section 6.3.84.1
0x00C0 + 1*n, n=0...191	RSA Public Key Prime	See section 6.3.84.2
0x0180 + 1*n, n=0...191	RSA Public Key Second	See section 6.3.84.3
0x0240	RSA ExponentL	See section 6.3.84.4
0x0241	RSA ExponentH	See section 6.3.84.5
0x0242 + 1*n, n=0...191	Encrypted SHA384 Hash	See section 6.3.84.6

6.3.84.1 RSA Public Key[n] (0x0000 + 1*n, n=0...191)

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.3.84.2 RSA Public Key Prime[n] (0x00C0 + 1*n, n=0...191)

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key Prime		

6.3.84.3 RSA Public Key Second[n] (0x0180 + 1*n, n=0...191)

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key Second		

6.3.84.4 RSA ExponentL - 0x0240

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentL	0x0	

6.3.84.5 RSA ExponentH - 0x0241

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentH	0x0	

6.3.84.6 Encrypted SHA384 Hash[n] (0x0242 + 1*n, n=0...191)

Bits	Field Name	Default NVM Value	Description
15:0	Encrypted SHA384 Hash	0x0	

6.3.85 Recovery FW Auth Header Section Summary Table

Word Offset	Description	Reference
0x0000	Device Blank NVM Device ID	See section 6.3.85.1
0x0001	Max Module AreaL	See section 6.3.85.2
0x0002	Max Module AreaH	See section 6.3.85.3
0x0003	Current Module AreaL	See section 6.3.85.4
0x0004	Current Module AreaH	See section 6.3.85.5
0x0005	Reserved	See section 6.3.85.6
0x0006	Code Revision	See section 6.3.85.7
0x0007	Reserved Spare Word	See section 6.3.85.8

6.3.85.1 Device Blank NVM Device ID - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Device Blank NVM Device ID	0x57AC	

6.3.85.2 Max Module AreaL - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaL	0x1000	

6.3.85.3 Max Module AreaH - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaH	0x0009	

6.3.85.4 Current Module AreaL - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaL	0xE000	

6.3.85.5 Current Module AreaH - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaH	0x0006	

6.3.85.6 Reserved - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	

6.3.85.7 Code Revision - 0x0006

Bits	Field Name	Default NVM Value	Description
15:8	Major Revision	0x0	
7:0	Minor Revision	0x0	

6.3.85.8 Reserved Spare Word - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	Reserved Spare Word	0x0	

6.3.86 Image Raw Data Section Summary Table

It contains the EMP processor code

Word Offset	Description	Reference
0x0000	EMP Image Raw Data	See section 6.3.86.1

6.3.86.1 EMP Image Raw Data - 0x0000

Raw data module length: 127648 words

6.3.87 EMP Global Module Section Summary Table

This section contains two sub-sections:

1. Vendor Specific Settings: Settings for external PHY or Modules. This part has a proprietary format in order to enable advanced, vendor specific, PHY setting. These settings are loaded into the external devices via the MDIO/I2C interface.
2. List of Qualified Modules: Parameter list of up to 16 modules. Per module list holds OUI, Revision and Version numbers.

Word Offset	Description	Reference
0x0000	Section Length	See section 6.3.87.1
0x0001	Number of Qualified Modules	See section 6.3.87.2
0x0002 + 12*n, n=0...15	Module OUI Bytes 0-1	See section 6.3.87.3
0x0003 + 12*n, n=0...15	Module OUI Byte 2	See section 6.3.87.4
0x0004 + 12*n, n=0...15	Vendor Part Number Bytes 0-1	See section 6.3.87.5
0x0005 + 12*n, n=0...15	Vendor Part Number Bytes 2-3	See section 6.3.87.6
0x0006 + 12*n, n=0...15	Vendor Part Number Bytes 4-5	See section 6.3.87.7
0x0007 + 12*n, n=0...15	Vendor Part Number Bytes 6-7	See section 6.3.87.8
0x0008 + 12*n, n=0...15	Vendor Part Number Bytes 8-9	See section 6.3.87.9
0x0009 + 12*n, n=0...15	Vendor Part Number Bytes 10-11	See section 6.3.87.10
0x000A + 12*n, n=0...15	Vendor Part Number Bytes 12-13	See section 6.3.87.11
0x000B + 12*n, n=0...15	Vendor Part Number Bytes 14-15	See section 6.3.87.12
0x000C + 12*n, n=0...15	Module Revision Number Bytes 0-1	See section 6.3.87.13
0x000D + 12*n, n=0...15	Module Revision Number Bytes 2-3	See section 6.3.87.14

6.3.87.1 Section Length - 0x0000

The length of the section in words. Note that section length does not include a count for the section length word.

Bits	Field Name	Default NVM Value	Description
15:0	Section Length		Length in: 2 Bytes unit - 1 First Section -> Word: EMP Global Module -> Section Length Last Section -> Word: EMP Global Module -> Module Revision Number Bytes 2-3

6.3.87.2 Number of Qualified Modules - 0x0001

Number of valid entries, 0 thru 15, in the qualified modules' list

Bits	Field Name	Default NVM Value	Description
15:0	Number of Qualified Modules	0x10	

6.3.87.3 Module OUI Bytes 0-1[n] (0x0002 + 12*n, n=0...15)

OUI of qualified external modules.

SFP Vendor OUI is located at offsets [39:37]

Bits	Field Name	Default NVM Value	Description
15:8	Byte 1	0x0	
7:0	Byte 0	0x0	

6.3.87.4 Module OUI Byte 2[n] (0x0003 + 12*n, n=0...15)

OUI of qualified external modules.

SFP Vendor OUI is located at offsets [39:37]

Bits	Field Name	Default NVM Value	Description
15:8	Reserved	0x0	
7:0	Byte 2	0x0	

6.3.87.5 Vendor Part Number Bytes 0-1[n] (0x0004 + 12*n, n=0...15)

Vendor Part Number of qualified external modules.

SFP+: Addr A0h Bytes 55:40

Bits	Field Name	Default NVM Value	Description
15:8	Byte1	0x0	
7:0	Byte0	0x0	

6.3.87.6 Vendor Part Number Bytes 2-3[n] (0x0005 + 12*n, n=0...15)

Vendor Part Number of qualified external modules.

SFP+: Addr A0h Bytes 55:40

Bits	Field Name	Default NVM Value	Description
15:8	Byte1	0x0	
7:0	Byte0	0x0	

6.3.87.7 Vendor Part Number Bytes 4-5[n] (0x0006 + 12*n, n=0...15)

Vendor Part Number of qualified external modules.

SFP+: Addr A0h Bytes 55:40

Bits	Field Name	Default NVM Value	Description
15:8	Byte1	0x0	
7:0	Byte0	0x0	

6.3.87.8 Vendor Part Number Bytes 6-7[n] (0x0007 + 12*n, n=0...15)

Vendor Part Number of qualified external modules.

SFP+: Addr A0h Bytes 55:40

Bits	Field Name	Default NVM Value	Description
15:8	Byte1	0x0	
7:0	Byte0	0x0	

6.3.87.9 Vendor Part Number Bytes 8-9[n] (0x0008 + 12*n, n=0...15)

Vendor Part Number of qualified external modules.

SFP+: Addr A0h Bytes 55:40

Bits	Field Name	Default NVM Value	Description
15:8	Byte1	0x0	
7:0	Byte0	0x0	

6.3.87.10 Vendor Part Number Bytes 10-11[n] (0x0009 + 12*n, n=0...15)

Vendor Part Number of qualified external modules.

SFP+: Addr A0h Bytes 55:40

Bits	Field Name	Default NVM Value	Description
15:8	Byte1	0x0	
7:0	Byte0	0x0	

6.3.87.11 Vendor Part Number Bytes 12-13[n] (0x000A + 12*n, n=0...15)

Vendor Part Number of qualified external modules.

SFP+: Addr A0h Bytes 55:40

Bits	Field Name	Default NVM Value	Description
15:8	Byte1	0x0	
7:0	Byte0	0x0	

6.3.87.12 Vendor Part Number Bytes 14-15[n] (0x000B + 12*n, n=0...15)

Vendor Part Number of qualified external modules.

SFP+: Addr A0h Bytes 55:40

Bits	Field Name	Default NVM Value	Description
15:8	Byte1	0x0	
7:0	Byte0	0x0	

6.3.87.13 Module Revision Number Bytes 0-1[n] (0x000C + 12*n, n=0...15)

Revision Number of qualified external modules.

SFP+: Addr A0h Bytes 59:56

Bits	Field Name	Default NVM Value	Description
15:8	Byte 1	0x0	
7:0	Byte 0	0x0	

6.3.87.14 Module Revision Number Bytes 2-3[n] (0x000D + 12*n, n=0...15)

Revision Number of qualified external modules.

SFP+: Addr A0h Bytes 59:56

Bits	Field Name	Default NVM Value	Description
15:8	Byte 1	0x0	
7:0	Byte 0	0x0	

6.3.88 EMP Settings Module Header Section Summary Table

This section contains the modes of operation of the EMP.

Word Offset	Description	Reference
0x0000	Section Length	See section 6.3.88.1
0x0001	Common Firmware Parameters	See section 6.3.88.2
0x0002	FW Misc	See section 6.3.88.3
0x0003	EEE Variables	See section 6.3.88.4
0x0004	Maximal Wear Out Value	See section 6.3.88.5
0x0005	Initial Wear Out Value	See section 6.3.88.6
0x0006	Staggering Delay	See section 6.3.88.7
0x0007	PXE PFC Timer Value	See section 6.3.88.8
0x0008	PXE GPC High Threshold Value	See section 6.3.88.9
0x0009	PXE GPC Low Threshold Value	See section 6.3.88.10
0x000A	Internal Thermal Sensor Maximum Secured Value	See section 6.3.88.11
0x000B	Internal Thermal Sensor Minimum Secured Value	See section 6.3.88.12
0x000C	MAX_LL_AQ_CREDITS	See section 6.3.88.13
0x000D	MAX_PWR_LMT	See section 6.3.88.14

6.3.88.1 Section Length - 0x0000

The length of the section in words. Note that section length does not include a count for the section length word.

Bits	Field Name	Default NVM Value	Description
15:0	Section Length		Length in: 2 Bytes unit - 1 First Section -> Word: EMP Settings Module Header -> Section Length Last Section -> Word: EMP Settings Module Header -> MAX_PWR_LMT

6.3.88.2 Common Firmware Parameters - 0x0001

Bits	Field Name	Default NVM Value	Description
15:10	Reserved	0x5	
9	PF reset on queue overflow	0x0	Valid values are: 0x0 = Disabled 0x1 = Enabled
8:0	Reserved	0x3	

6.3.88.3 FW Misc - 0x0002

Bits	Field Name	Default NVM Value	Description
15:1	FW Reserved	0x0	Reserved
0	VEB Statistics Disable	0x0	When clear (Default) VEB Statistics are Enable. When set VEB Statistics are Disabled.

6.3.88.4 EEE Variables - 0x0003

Bits	Field Name	Default NVM Value	Description
15:8	receiveTW	0xe	Value determines the time (expressed in microseconds) that the receiving link partner is requesting the transmitting link partner to wait before starting the transmission data following the LPI. This value is platform dependent and depends on the OEM platform.
7:0	transmitTW	0xe	Value determines the time (expressed in microseconds) that the transmitting link partner will wait before it starts transmitting data after leaving the Low Power Idle (LPI) mode. This value is platform dependent and depends on the OEM platform.

6.3.88.5 Maximal Wear Out Value - 0x0004

Maximal wear out credit value

Bits	Field Name	Default NVM Value	Description
15:0	MAX	0x1400	

6.3.88.6 Initial Wear Out Value - 0x0005

Initial wear out credit value

Bits	Field Name	Default NVM Value	Description
15:0	INITIAL	0x800	

6.3.88.7 Staggering Delay - 0x0006

Port staggering Delay

Bits	Field Name	Default NVM Value	Description
15:0	Delay	0x0	Port staggering delay in usec.

6.3.88.8 PXE PFC Timer Value - 0x0007

Bits	Field Name	Default NVM Value	Description
15:14	Reserved	0x0	Reserved.
13:0	PXE PFC Timer	0x4	The value [in ms] to be set as the PFC timer in PCE mode.

6.3.88.9 PXE GPC High Threshold Value - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	PXE GPC High Threshold	0x0080	High threshold [in quanta of 32B] of GPC to be used in the RPB monitoring flow.

6.3.88.10 PXE GPC Low Threshold Value - 0x0009

Bits	Field Name	Default NVM Value	Description
15:0	PXE GPC Low Threshold Value	0x0010	Low threshold [in quanta of 32B] of GPC to be used in the RPB monitoring flow.

6.3.88.11 Internal Thermal Sensor Maximum Secured Value - 0x000A

Bits	Field Name	Default NVM Value	Description
15:0	Internal Thermal Sensor Maximum Secured Value	0x73	0-255 Recommended 0-115 degrees

6.3.88.12 Internal Thermal Sensor Minimum Secured Value - 0x000B

Bits	Field Name	Default NVM Value	Description
15:0	Internal Thermal Sensor Minimum Secured Value	0x60	0-255 Recommended 90-100 degrees

6.3.88.13 MAX_LL_AQ_CREDITS - 0x000C

Bits	Field Name	Default NVM Value	Description
15:0	MAX_LL_AQ_CREDITS	0x8	

6.3.88.14 MAX_PWR_LMT - 0x000D

Bits	Field Name	Default NVM Value	Description
15:8	Board max power limit	0xE	Maximum allowed power for board. 0.5W resolution
7:0	Cage max power limit	0x7	Maximum allowed power for any cage. 0.5W resolution

6.3.89 Allow list Section Summary Table

6.3.90 Configuration Metadata Section Summary Table

NVM Adaptive Configuration Metadata

Word Offset	Description	Reference

6.3.91 DL Scripts Section Summary Table

6.3.92 Analog PHY Configuration Section Summary Table

Includes static configurations of SerDes PHY registers for link establishment

Word Offset	Description	Reference
0x0000	Section Length low	See section 6.3.92.1
0x0001	Section Length high	See section 6.3.92.2
0x0002	Reg Write Indirect List	See section 6.3.92.3
0x0003	Reg Write Indirect List 2	See section 6.3.92.4

6.3.92.1 Section Length low - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Section Length low	0x00000000	Length in: 2 Bytes unit - 2 First Section -> Word: Analog PHY Configuration -> Section Length low Last Section -> Word: Analog PHY Configuration -> Reg Write Indirect List 2

6.3.92.2 Section Length high - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Section Length high		

6.3.92.3 Reg Write Indirect List - 0x0002

Raw data module length: variable

6.3.92.4 Reg Write Indirect List 2 - 0x0003

Raw data module length: variable

6.3.93 EMP Image Section Summary Table

This module contains the EMP processor code

Word Offset	Description	Reference
0x0000	EMP Image Raw Data	See section 6.3.93.1

6.3.93.1 EMP Image Raw Data - 0x0000

Raw data module length: variable

6.3.94 RDE Dictionaries Section Summary Table

Word Offset	Description	Reference
0x0000	Dictionaries	See section 6.3.94.1

6.3.94.1 Dictionaries - 0x0000

Raw data module length: variable

6.3.95 External Topology Device Image 0 Section Summary Table

Word Offset	Description	Reference
0x0000	Section Length Low	See section 6.3.95.1
0x0001	Section Length High	See section 6.3.95.2
0x0002	Topology Device Image Version Number High 0	See section 6.3.95.3
0x0003	Topology Device Image Version Number High 1	See section 6.3.95.4
0x0004	Topology Device Image Version Number Low 0	See section 6.3.95.5
0x0005	Topology Device Image Version Number Low 1	See section 6.3.95.6
0x0006	Flags and Device Part Number	See section 6.3.95.7
0x0007	Number of Sub-Sections	See section 6.3.95.8
0x0008	External Topology Device Image	See section 6.3.95.9

6.3.95.1 Section Length Low - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Section Length Low		Length in: 2 Bytes unit - 2 First Section -> Word: External Topology Device Image 0 -> Section Length Low Last Section -> Word: External Topology Device Image 0 -> External Topology Device Image

6.3.95.2 Section Length High - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Section Length High	0x0	

6.3.95.3 Topology Device Image Version Number High 0 - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	Topology Device Image Version Number High 0	0x0	

6.3.95.4 Topology Device Image Version Number High 1 - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	Topology Device Image Version Number High 1	0x0	

6.3.95.5 Topology Device Image Version Number Low 0 - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	Topology Device Image Version Number Low 0	0x0	

6.3.95.6 Topology Device Image Version Number Low 1 - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	Topology Device Image Version Number Low 1	0x0	

6.3.95.7 Flags and Device Part Number - 0x0006

Bits	Field Name	Default NVM Value	Description
15:8	Device Part Number	0x0	0x31 for Parkvale 0x32 for Coppervale - 4 ports (X557-AT4) 0x37 for Coppervale - 2 ports (X557-AT2)
7:2	Reserved	0x0	
1	ProgEn	0x0	
0	LoadEn	0x0	

6.3.95.8 Number of Sub-Sections - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	Number of Sub-Sections	0x0	

6.3.95.9 External Topology Device Image - 0x0008

Raw data module length: variable

6.3.96 External Topology Device Image 1 Section Summary Table

Word Offset	Description	Reference
0x0000	Section Length Low	See section 6.3.96.1
0x0001	Section Length High	See section 6.3.96.2
0x0002	Topology Device Image Version Number High 0	See section 6.3.96.3
0x0003	Topology Device Image Version Number High 1	See section 6.3.96.4
0x0004	Topology Device Image Version Number Low 0	See section 6.3.96.5
0x0005	Topology Device Image Version Number Low 1	See section 6.3.96.6
0x0006	Flags and Device Part Number	See section 6.3.96.7
0x0007	Number of Sub-Sections	See section 6.3.96.8
0x0008	External Topology Device Image	See section 6.3.96.9

6.3.96.1 Section Length Low - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Section Length Low		Length in: 2 Bytes unit - 2 First Section -> Word: External Topology Device Image 1 -> Section Length Low Last Section -> Word: External Topology Device Image 1 -> External Topology Device Image

6.3.96.2 Section Length High - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Section Length High	0x0	

6.3.96.3 Topology Device Image Version Number High 0 - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	Topology Device Image Version Number High 0	0x0	

6.3.96.4 Topology Device Image Version Number High 1 - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	Topology Device Image Version Number High 1	0x0	

6.3.96.5 Topology Device Image Version Number Low 0 - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	Topology Device Image Version Number Low 0	0x0	

6.3.96.6 Topology Device Image Version Number Low 1 - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	Topology Device Image Version Number Low 1	0x0	

6.3.96.7 Flags and Device Part Number - 0x0006

Bits	Field Name	Default NVM Value	Description
15:8	Device Part Number	0x0	0x31 for Parkvale 0x32 for Coppervale - 4 ports (X557-AT4) 0x37 for Coppervale - 2 ports (X557-AT2)
7:2	Reserved	0x0	
1	ProgEn	0x0	
0	LoadEn	0x0	

6.3.96.8 Number of Sub-Sections - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	Number of Sub-Sections	0x0	

6.3.96.9 External Topology Device Image - 0x0008

Raw data module length: variable

6.3.97 External Topology Device Image 2 Section Summary Table

Word Offset	Description	Reference
0x0000	Section Length Low	See section 6.3.97.1
0x0001	Section Length High	See section 6.3.97.2
0x0002	Topology Device Image Version Number High 0	See section 6.3.97.3
0x0003	Topology Device Image Version Number High 1	See section 6.3.97.4
0x0004	Topology Device Image Version Number Low 0	See section 6.3.97.5
0x0005	Topology Device Image Version Number Low 1	See section 6.3.97.6
0x0006	Flags and Device Part Number	See section 6.3.97.7
0x0007	Number of Sub-Sections	See section 6.3.97.8
0x0008	External Topology Device Image	See section 6.3.97.9

6.3.97.1 Section Length Low - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Section Length Low		Length in: 2 Bytes unit - 2 First Section -> Word: External Topology Device Image 2 -> Section Length Low Last Section -> Word: External Topology Device Image 2 -> External Topology Device Image

6.3.97.2 Section Length High - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Section Length High	0x0	

6.3.97.3 Topology Device Image Version Number High 0 - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	Topology Device Image Version Number High 0	0x0	

6.3.97.4 Topology Device Image Version Number High 1 - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	Topology Device Image Version Number High 1	0x0	

6.3.97.5 Topology Device Image Version Number Low 0 - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	Topology Device Image Version Number Low 0	0x0	

6.3.97.6 Topology Device Image Version Number Low 1 - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	Topology Device Image Version Number Low 1	0x0	

6.3.97.7 Flags and Device Part Number - 0x0006

Bits	Field Name	Default NVM Value	Description
15:8	Device Part Number	0x0	0x31 for Parkvale 0x32 for Coppervale - 4 ports (X557-AT4) 0x37 for Coppervale - 2 ports (X557-AT2)
7:2	Reserved	0x0	
1	ProgEn	0x0	
0	LoadEn	0x0	

6.3.97.8 Number of Sub-Sections - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	Number of Sub-Sections	0x0	

6.3.97.9 External Topology Device Image - 0x0008

Raw data module length: variable

6.3.98 External Topology Device Image 3 Section Summary Table

Word Offset	Description	Reference
0x0000	Section Length Low	See section 6.3.98.1
0x0001	Section Length High	See section 6.3.98.2
0x0002	Topology Device Image Version Number High 0	See section 6.3.98.3
0x0003	Topology Device Image Version Number High 1	See section 6.3.98.4
0x0004	Topology Device Image Version Number Low 0	See section 6.3.98.5
0x0005	Topology Device Image Version Number Low 1	See section 6.3.98.6
0x0006	Flags and Device Part Number	See section 6.3.98.7
0x0007	Number of Sub-Sections	See section 6.3.98.8
0x0008	External Topology Device Image	See section 6.3.98.9

6.3.98.1 Section Length Low - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Section Length Low		Length in: 2 Bytes unit - 2 First Section -> Word: External Topology Device Image 3 -> Section Length Low Last Section -> Word: External Topology Device Image 3 -> External Topology Device Image

6.3.98.2 Section Length High - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Section Length High	0x0	

6.3.98.3 Topology Device Image Version Number High 0 - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	Topology Device Image Version Number High 0	0x0	

6.3.98.4 Topology Device Image Version Number High 1 - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	Topology Device Image Version Number High 1	0x0	

6.3.98.5 Topology Device Image Version Number Low 0 - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	Topology Device Image Version Number Low 0	0x0	

6.3.98.6 Topology Device Image Version Number Low 1 - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	Topology Device Image Version Number Low 1	0x0	

6.3.98.7 Flags and Device Part Number - 0x0006

Bits	Field Name	Default NVM Value	Description
15:8	Device Part Number	0x0	0x31 for Parkvale 0x32 for Coppervale - 4 ports (X557-AT4) 0x37 for Coppervale - 2 ports (X557-AT2)
7:2	Reserved	0x0	
1	ProgEn	0x0	
0	LoadEn	0x0	

6.3.98.8 Number of Sub-Sections - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	Number of Sub-Sections	0x0	

6.3.98.9 External Topology Device Image - 0x0008

Raw data module length: variable

6.3.99 NVM Hash Module Section Summary Table

This section contains the Page wise hash values of the NVM sections which would be read after the NVM is authenticated

Word Offset	Description	Reference
0x0000	Start Section Pointer	See section 6.3.99.1
0x0001	End Section Pointer	See section 6.3.99.2
0x0002 + 1*n, n=0...23	Sector 0 SHA384 Hash	See section 6.3.99.3
0x001A + 1*n, n=0...23	Sector 1 SHA384 Hash	See section 6.3.99.4
0x0032 + 1*n, n=0...23	Sector 2 SHA384 Hash	See section 6.3.99.5
0x004A + 1*n, n=0...23	Sector 3 SHA384 Hash	See section 6.3.99.6
0x0062 + 1*n, n=0...23	Sector 4 SHA384 Hash	See section 6.3.99.7
0x007A + 1*n, n=0...23	Sector 5 SHA384 Hash	See section 6.3.99.8
0x0092 + 1*n, n=0...23	Sector 6 SHA384 Hash	See section 6.3.99.9
0x00AA + 1*n, n=0...23	Sector 7 SHA384 Hash	See section 6.3.99.10
0x00C2 + 1*n, n=0...23	Sector 8 SHA384 Hash	See section 6.3.99.11
0x00DA + 1*n, n=0...23	Sector 9 SHA384 Hash	See section 6.3.99.12
0x00F2 + 1*n, n=0...23	Sector 10 SHA384 Hash	See section 6.3.99.13
0x010A + 1*n, n=0...23	Sector 11 SHA384 Hash	See section 6.3.99.14
0x0122 + 1*n, n=0...23	Sector 12 SHA384 Hash	See section 6.3.99.15
0x013A + 1*n, n=0...23	Sector 13 SHA384 Hash	See section 6.3.99.16
0x0152 + 1*n, n=0...23	Sector 14 SHA384 Hash	See section 6.3.99.17
0x016A + 1*n, n=0...23	Sector 15 SHA384 Hash	See section 6.3.99.18
0x0182 + 1*n, n=0...23	Sector 16 SHA384 Hash	See section 6.3.99.19
0x019A + 1*n, n=0...23	Sector 17 SHA384 Hash	See section 6.3.99.20
0x01B2 + 1*n, n=0...23	Sector 18 SHA384 Hash	See section 6.3.99.21
0x01CA + 1*n, n=0...23	Sector 19 SHA384 Hash	See section 6.3.99.22
0x01E2 + 1*n, n=0...23	Sector 20 SHA384 Hash	See section 6.3.99.23
0x01FA + 1*n, n=0...23	Sector 21 SHA384 Hash	See section 6.3.99.24
0x0212 + 1*n, n=0...23	Sector 22 SHA384 Hash	See section 6.3.99.25
0x022A + 1*n, n=0...23	Sector 23 SHA384 Hash	See section 6.3.99.26
0x0242 + 1*n, n=0...23	Sector 24 SHA384 Hash	See section 6.3.99.27
0x025A + 1*n, n=0...23	Sector 25 SHA384 Hash	See section 6.3.99.28
0x0272 + 1*n, n=0...23	Sector 26 SHA384 Hash	See section 6.3.99.29
0x028A + 1*n, n=0...23	Sector 27 SHA384 Hash	See section 6.3.99.30
0x02A2 + 1*n, n=0...23	Sector 28 SHA384 Hash	See section 6.3.99.31
0x02BA + 1*n, n=0...23	Sector 29 SHA384 Hash	See section 6.3.99.32
0x02D2 + 1*n, n=0...23	Sector 30 SHA384 Hash	See section 6.3.99.33
0x02EA + 1*n, n=0...23	Sector 31 SHA384 Hash	See section 6.3.99.34
0x0302 + 1*n, n=0...23	Sector 32 SHA384 Hash	See section 6.3.99.35

Word Offset	Description	Reference
0x031A + 1*n, n=0...23	Sector 33 SHA384 Hash	See section 6.3.99.36
0x0332 + 1*n, n=0...23	Sector 34 SHA384 Hash	See section 6.3.99.37
0x034A + 1*n, n=0...23	Sector 35 SHA384 Hash	See section 6.3.99.38
0x0362 + 1*n, n=0...23	Sector 36 SHA384 Hash	See section 6.3.99.39
0x037A + 1*n, n=0...23	Sector 37 SHA384 Hash	See section 6.3.99.40
0x0392 + 1*n, n=0...23	Sector 38 SHA384 Hash	See section 6.3.99.41
0x03AA + 1*n, n=0...23	Sector 39 SHA384 Hash	See section 6.3.99.42
0x03C2 + 1*n, n=0...23	Sector 40 SHA384 Hash	See section 6.3.99.43
0x03DA + 1*n, n=0...23	Sector 41 SHA384 Hash	See section 6.3.99.44
0x03F2 + 1*n, n=0...23	Sector 42 SHA384 Hash	See section 6.3.99.45
0x040A + 1*n, n=0...23	Sector 43 SHA384 Hash	See section 6.3.99.46
0x0422 + 1*n, n=0...23	Sector 44 SHA384 Hash	See section 6.3.99.47
0x043A + 1*n, n=0...23	Sector 45 SHA384 Hash	See section 6.3.99.48
0x0452 + 1*n, n=0...23	Sector 46 SHA384 Hash	See section 6.3.99.49
0x046A + 1*n, n=0...23	Sector 47 SHA384 Hash	See section 6.3.99.50
0x0482 + 1*n, n=0...23	Sector 48 SHA384 Hash	See section 6.3.99.51
0x049A + 1*n, n=0...23	Sector 49 SHA384 Hash	See section 6.3.99.52
0x04B2 + 1*n, n=0...23	Sector 50 SHA384 Hash	See section 6.3.99.53
0x04CA + 1*n, n=0...23	Sector 51 SHA384 Hash	See section 6.3.99.54
0x04E2 + 1*n, n=0...23	Sector 52 SHA384 Hash	See section 6.3.99.55
0x04FA + 1*n, n=0...23	Sector 53 SHA384 Hash	See section 6.3.99.56
0x0512 + 1*n, n=0...23	Sector 54 SHA384 Hash	See section 6.3.99.57
0x052A + 1*n, n=0...23	Sector 55 SHA384 Hash	See section 6.3.99.58
0x0542 + 1*n, n=0...23	Sector 56 SHA384 Hash	See section 6.3.99.59
0x055A + 1*n, n=0...23	Sector 57 SHA384 Hash	See section 6.3.99.60
0x0572 + 1*n, n=0...23	Sector 58 SHA384 Hash	See section 6.3.99.61
0x058A + 1*n, n=0...23	Sector 59 SHA384 Hash	See section 6.3.99.62
0x05A2 + 1*n, n=0...23	Sector 60 SHA384 Hash	See section 6.3.99.63
0x05BA + 1*n, n=0...23	Sector 61 SHA384 Hash	See section 6.3.99.64
0x05D2 + 1*n, n=0...23	Sector 62 SHA384 Hash	See section 6.3.99.65
0x05EA + 1*n, n=0...23	Sector 63 SHA384 Hash	See section 6.3.99.66
0x0602 + 1*n, n=0...23	Sector 64 SHA384 Hash	See section 6.3.99.67
0x061A + 1*n, n=0...23	Sector 65 SHA384 Hash	See section 6.3.99.68
0x0632 + 1*n, n=0...23	Sector 66 SHA384 Hash	See section 6.3.99.69
0x064A + 1*n, n=0...23	Sector 67 SHA384 Hash	See section 6.3.99.70
0x0662 + 1*n, n=0...23	Sector 68 SHA384 Hash	See section 6.3.99.71
0x067A + 1*n, n=0...23	Sector 69 SHA384 Hash	See section 6.3.99.72
0x0692 + 1*n, n=0...23	Sector 70 SHA384 Hash	See section 6.3.99.73
0x06AA + 1*n, n=0...23	Sector 71 SHA384 Hash	See section 6.3.99.74

Word Offset	Description	Reference
0x06C2 + 1*n, n=0...23	Sector 72 SHA384 Hash	See section 6.3.99.75
0x06DA + 1*n, n=0...23	Sector 73 SHA384 Hash	See section 6.3.99.76
0x06F2 + 1*n, n=0...23	Sector 74 SHA384 Hash	See section 6.3.99.77
0x070A + 1*n, n=0...23	Sector 75 SHA384 Hash	See section 6.3.99.78
0x0722 + 1*n, n=0...23	Sector 76 SHA384 Hash	See section 6.3.99.79
0x073A + 1*n, n=0...23	Sector 77 SHA384 Hash	See section 6.3.99.80
0x0752 + 1*n, n=0...23	Sector 78 SHA384 Hash	See section 6.3.99.81
0x076A + 1*n, n=0...23	Sector 79 SHA384 Hash	See section 6.3.99.82
0x0782 + 1*n, n=0...23	Sector 80 SHA384 Hash	See section 6.3.99.83
0x079A + 1*n, n=0...23	Sector 81 SHA384 Hash	See section 6.3.99.84
0x07B2 + 1*n, n=0...23	Sector 82 SHA384 Hash	See section 6.3.99.85
0x07CA + 1*n, n=0...23	Sector 83 SHA384 Hash	See section 6.3.99.86
0x07E2 + 1*n, n=0...23	Sector 84 SHA384 Hash	See section 6.3.99.87
0x07FA + 1*n, n=0...23	Sector 85 SHA384 Hash	See section 6.3.99.88
0x0812 + 1*n, n=0...23	Sector 86 SHA384 Hash	See section 6.3.99.89
0x082A + 1*n, n=0...23	Sector 87 SHA384 Hash	See section 6.3.99.90
0x0842 + 1*n, n=0...23	Sector 88 SHA384 Hash	See section 6.3.99.91
0x085A + 1*n, n=0...23	Sector 89 SHA384 Hash	See section 6.3.99.92
0x0872 + 1*n, n=0...23	Sector 90 SHA384 Hash	See section 6.3.99.93
0x088A + 1*n, n=0...23	Sector 91 SHA384 Hash	See section 6.3.99.94
0x08A2 + 1*n, n=0...23	Sector 92 SHA384 Hash	See section 6.3.99.95
0x08BA + 1*n, n=0...23	Sector 93 SHA384 Hash	See section 6.3.99.96
0x08D2 + 1*n, n=0...23	Sector 94 SHA384 Hash	See section 6.3.99.97
0x08EA + 1*n, n=0...23	Sector 95 SHA384 Hash	See section 6.3.99.98
0x0902 + 1*n, n=0...23	Sector 96 SHA384 Hash	See section 6.3.99.99
0x091A + 1*n, n=0...23	Sector 97 SHA384 Hash	See section 6.3.99.100
0x0932 + 1*n, n=0...23	Sector 98 SHA384 Hash	See section 6.3.99.101
0x094A + 1*n, n=0...23	Sector 99 SHA384 Hash	See section 6.3.99.102
0x0962 + 1*n, n=0...23	Sector 100 SHA384 Hash	See section 6.3.99.103
0x097A + 1*n, n=0...23	Sector 101 SHA384 Hash	See section 6.3.99.104
0x0992 + 1*n, n=0...23	Sector 102 SHA384 Hash	See section 6.3.99.105
0x09AA + 1*n, n=0...23	Sector 103 SHA384 Hash	See section 6.3.99.106
0x09C2 + 1*n, n=0...23	Sector 104 SHA384 Hash	See section 6.3.99.107
0x09DA + 1*n, n=0...23	Sector 105 SHA384 Hash	See section 6.3.99.108
0x09F2 + 1*n, n=0...23	Sector 106 SHA384 Hash	See section 6.3.99.109
0x0A0A + 1*n, n=0...23	Sector 107 SHA384 Hash	See section 6.3.99.110
0x0A22 + 1*n, n=0...23	Sector 108 SHA384 Hash	See section 6.3.99.111
0x0A3A + 1*n, n=0...23	Sector 109 SHA384 Hash	See section 6.3.99.112
0x0A52 + 1*n, n=0...23	Sector 110 SHA384 Hash	See section 6.3.99.113

Word Offset	Description	Reference
0x0A6A + 1*n, n=0...23	Sector 111 SHA384 Hash	See section 6.3.99.114
0x0A82 + 1*n, n=0...23	Sector 112 SHA384 Hash	See section 6.3.99.115
0x0A9A + 1*n, n=0...23	Sector 113 SHA384 Hash	See section 6.3.99.116
0x0AB2 + 1*n, n=0...23	Sector 114 SHA384 Hash	See section 6.3.99.117
0x0ACA + 1*n, n=0...23	Sector 115 SHA384 Hash	See section 6.3.99.118
0x0AE2 + 1*n, n=0...23	Sector 116 SHA384 Hash	See section 6.3.99.119
0x0AFA + 1*n, n=0...23	Sector 117 SHA384 Hash	See section 6.3.99.120
0x0B12 + 1*n, n=0...23	Sector 118 SHA384 Hash	See section 6.3.99.121
0x0B2A + 1*n, n=0...23	Sector 119 SHA384 Hash	See section 6.3.99.122
0x0B42 + 1*n, n=0...23	Sector 120 SHA384 Hash	See section 6.3.99.123
0x0B5A + 1*n, n=0...23	Sector 121 SHA384 Hash	See section 6.3.99.124
0x0B72 + 1*n, n=0...23	Sector 122 SHA384 Hash	See section 6.3.99.125
0x0B8A + 1*n, n=0...23	Sector 123 SHA384 Hash	See section 6.3.99.126
0x0BA2 + 1*n, n=0...23	Sector 124 SHA384 Hash	See section 6.3.99.127
0x0BBA + 1*n, n=0...23	Sector 125 SHA384 Hash	See section 6.3.99.128
0x0BD2 + 1*n, n=0...23	Sector 126 SHA384 Hash	See section 6.3.99.129
0x0BEA + 1*n, n=0...23	Sector 127 SHA384 Hash	See section 6.3.99.130
0x0C02 + 1*n, n=0...23	Sector 128 SHA384 Hash	See section 6.3.99.131
0x0C1A + 1*n, n=0...23	Sector 129 SHA384 Hash	See section 6.3.99.132
0x0C32 + 1*n, n=0...23	Sector 130 SHA384 Hash	See section 6.3.99.133
0x0C4A + 1*n, n=0...23	Sector 131 SHA384 Hash	See section 6.3.99.134
0x0C62 + 1*n, n=0...23	Sector 132 SHA384 Hash	See section 6.3.99.135
0x0C7A + 1*n, n=0...23	Sector 133 SHA384 Hash	See section 6.3.99.136
0x0C92 + 1*n, n=0...23	Sector 134 SHA384 Hash	See section 6.3.99.137
0x0CAA + 1*n, n=0...23	Sector 135 SHA384 Hash	See section 6.3.99.138
0x0CC2 + 1*n, n=0...23	Sector 136 SHA384 Hash	See section 6.3.99.139
0x0CDA + 1*n, n=0...23	Sector 137 SHA384 Hash	See section 6.3.99.140
0x0CF2 + 1*n, n=0...23	Sector 138 SHA384 Hash	See section 6.3.99.141
0x0D0A + 1*n, n=0...23	Sector 139 SHA384 Hash	See section 6.3.99.142
0x0D22 + 1*n, n=0...23	Sector 140 SHA384 Hash	See section 6.3.99.143
0x0D3A + 1*n, n=0...23	Sector 141 SHA384 Hash	See section 6.3.99.144
0x0D52 + 1*n, n=0...23	Sector 142 SHA384 Hash	See section 6.3.99.145
0x0D6A + 1*n, n=0...23	Sector 143 SHA384 Hash	See section 6.3.99.146
0x0D82 + 1*n, n=0...23	Sector 144 SHA384 Hash	See section 6.3.99.147
0x0D9A + 1*n, n=0...23	Sector 145 SHA384 Hash	See section 6.3.99.148
0x0DB2 + 1*n, n=0...23	Sector 146 SHA384 Hash	See section 6.3.99.149
0x0DCA + 1*n, n=0...23	Sector 147 SHA384 Hash	See section 6.3.99.150
0x0DE2 + 1*n, n=0...23	Sector 148 SHA384 Hash	See section 6.3.99.151
0x0DFA + 1*n, n=0...23	Sector 149 SHA384 Hash	See section 6.3.99.152

Word Offset	Description	Reference
0x0E12 + 1*n, n=0...23	Sector 150 SHA384 Hash	See section 6.3.99.153
0x0E2A + 1*n, n=0...23	Sector 151 SHA384 Hash	See section 6.3.99.154
0x0E42 + 1*n, n=0...23	Sector 152 SHA384 Hash	See section 6.3.99.155
0x0E5A + 1*n, n=0...23	Sector 153 SHA384 Hash	See section 6.3.99.156
0x0E72 + 1*n, n=0...23	Sector 154 SHA384 Hash	See section 6.3.99.157
0x0E8A + 1*n, n=0...23	Sector 155 SHA384 Hash	See section 6.3.99.158
0x0EA2 + 1*n, n=0...23	Sector 156 SHA384 Hash	See section 6.3.99.159
0x0EBA + 1*n, n=0...23	Sector 157 SHA384 Hash	See section 6.3.99.160
0x0ED2 + 1*n, n=0...23	Sector 158 SHA384 Hash	See section 6.3.99.161
0x0EEA + 1*n, n=0...23	Sector 159 SHA384 Hash	See section 6.3.99.162
0x0F02 + 1*n, n=0...23	Sector 160 SHA384 Hash	See section 6.3.99.163
0x0F1A + 1*n, n=0...23	Sector 161 SHA384 Hash	See section 6.3.99.164
0x0F32 + 1*n, n=0...23	Sector 162 SHA384 Hash	See section 6.3.99.165
0x0F4A + 1*n, n=0...23	Sector 163 SHA384 Hash	See section 6.3.99.166
0x0F62 + 1*n, n=0...23	Sector 164 SHA384 Hash	See section 6.3.99.167
0x0F7A + 1*n, n=0...23	Sector 165 SHA384 Hash	See section 6.3.99.168
0x0F92 + 1*n, n=0...23	Sector 166 SHA384 Hash	See section 6.3.99.169
0x0FAA + 1*n, n=0...23	Sector 167 SHA384 Hash	See section 6.3.99.170
0x0FC2 + 1*n, n=0...23	Sector 168 SHA384 Hash	See section 6.3.99.171
0x0FDA + 1*n, n=0...23	Sector 169 SHA384 Hash	See section 6.3.99.172
0x0FF2 + 1*n, n=0...23	Sector 170 SHA384 Hash	See section 6.3.99.173
0x100A + 1*n, n=0...23	Sector 171 SHA384 Hash	See section 6.3.99.174
0x1022 + 1*n, n=0...23	Sector 172 SHA384 Hash	See section 6.3.99.175
0x103A + 1*n, n=0...23	Sector 173 SHA384 Hash	See section 6.3.99.176
0x1052 + 1*n, n=0...23	Sector 174 SHA384 Hash	See section 6.3.99.177
0x106A + 1*n, n=0...23	Sector 175 SHA384 Hash	See section 6.3.99.178
0x1082 + 1*n, n=0...23	Sector 176 SHA384 Hash	See section 6.3.99.179
0x109A + 1*n, n=0...23	Sector 177 SHA384 Hash	See section 6.3.99.180
0x10B2 + 1*n, n=0...23	Sector 178 SHA384 Hash	See section 6.3.99.181
0x10CA + 1*n, n=0...23	Sector 179 SHA384 Hash	See section 6.3.99.182
0x10E2 + 1*n, n=0...23	Sector 180 SHA384 Hash	See section 6.3.99.183
0x10FA + 1*n, n=0...23	Sector 181 SHA384 Hash	See section 6.3.99.184
0x1112 + 1*n, n=0...23	Sector 182 SHA384 Hash	See section 6.3.99.185
0x112A + 1*n, n=0...23	Sector 183 SHA384 Hash	See section 6.3.99.186
0x1142 + 1*n, n=0...23	Sector 184 SHA384 Hash	See section 6.3.99.187
0x115A + 1*n, n=0...23	Sector 185 SHA384 Hash	See section 6.3.99.188
0x1172 + 1*n, n=0...23	Sector 186 SHA384 Hash	See section 6.3.99.189
0x118A + 1*n, n=0...23	Sector 187 SHA384 Hash	See section 6.3.99.190
0x11A2 + 1*n, n=0...23	Sector 188 SHA384 Hash	See section 6.3.99.191

Word Offset	Description	Reference
0x11BA + 1*n, n=0...23	Sector 189 SHA384 Hash	See section 6.3.99.192
0x11D2 + 1*n, n=0...23	Sector 190 SHA384 Hash	See section 6.3.99.193
0x11EA + 1*n, n=0...23	Sector 191 SHA384 Hash	See section 6.3.99.194
0x1202 + 1*n, n=0...23	Sector 192 SHA384 Hash	See section 6.3.99.195
0x121A + 1*n, n=0...23	Sector 193 SHA384 Hash	See section 6.3.99.196
0x1232 + 1*n, n=0...23	Sector 194 SHA384 Hash	See section 6.3.99.197
0x124A + 1*n, n=0...23	Sector 195 SHA384 Hash	See section 6.3.99.198
0x1262 + 1*n, n=0...23	Sector 196 SHA384 Hash	See section 6.3.99.199
0x127A + 1*n, n=0...23	Sector 197 SHA384 Hash	See section 6.3.99.200
0x1292 + 1*n, n=0...23	Sector 198 SHA384 Hash	See section 6.3.99.201
0x12AA + 1*n, n=0...23	Sector 199 SHA384 Hash	See section 6.3.99.202
0x12C2 + 1*n, n=0...23	Sector 200 SHA384 Hash	See section 6.3.99.203
0x12DA + 1*n, n=0...23	Sector 201 SHA384 Hash	See section 6.3.99.204
0x12F2 + 1*n, n=0...23	Sector 202 SHA384 Hash	See section 6.3.99.205
0x130A + 1*n, n=0...23	Sector 203 SHA384 Hash	See section 6.3.99.206
0x1322 + 1*n, n=0...23	Sector 204 SHA384 Hash	See section 6.3.99.207
0x133A + 1*n, n=0...23	Sector 205 SHA384 Hash	See section 6.3.99.208
0x1352 + 1*n, n=0...23	Sector 206 SHA384 Hash	See section 6.3.99.209
0x136A + 1*n, n=0...23	Sector 207 SHA384 Hash	See section 6.3.99.210
0x1382 + 1*n, n=0...23	Sector 208 SHA384 Hash	See section 6.3.99.211
0x139A + 1*n, n=0...23	Sector 209 SHA384 Hash	See section 6.3.99.212
0x13B2 + 1*n, n=0...23	Sector 210 SHA384 Hash	See section 6.3.99.213
0x13CA + 1*n, n=0...23	Sector 211 SHA384 Hash	See section 6.3.99.214
0x13E2 + 1*n, n=0...23	Sector 212 SHA384 Hash	See section 6.3.99.215
0x13FA + 1*n, n=0...23	Sector 213 SHA384 Hash	See section 6.3.99.216
0x1412 + 1*n, n=0...23	Sector 214 SHA384 Hash	See section 6.3.99.217
0x142A + 1*n, n=0...23	Sector 215 SHA384 Hash	See section 6.3.99.218
0x1442 + 1*n, n=0...23	Sector 216 SHA384 Hash	See section 6.3.99.219
0x145A + 1*n, n=0...23	Sector 217 SHA384 Hash	See section 6.3.99.220
0x1472 + 1*n, n=0...23	Sector 218 SHA384 Hash	See section 6.3.99.221
0x148A + 1*n, n=0...23	Sector 219 SHA384 Hash	See section 6.3.99.222
0x14A2 + 1*n, n=0...23	Sector 220 SHA384 Hash	See section 6.3.99.223
0x14BA + 1*n, n=0...23	Sector 221 SHA384 Hash	See section 6.3.99.224
0x14D2 + 1*n, n=0...23	Sector 222 SHA384 Hash	See section 6.3.99.225
0x14EA + 1*n, n=0...23	Sector 223 SHA384 Hash	See section 6.3.99.226
0x1502 + 1*n, n=0...23	Sector 224 SHA384 Hash	See section 6.3.99.227
0x151A + 1*n, n=0...23	Sector 225 SHA384 Hash	See section 6.3.99.228
0x1532 + 1*n, n=0...23	Sector 226 SHA384 Hash	See section 6.3.99.229
0x154A + 1*n, n=0...23	Sector 227 SHA384 Hash	See section 6.3.99.230

Word Offset	Description	Reference
0x1562 + 1*n, n=0...23	Sector 228 SHA384 Hash	See section 6.3.99.231
0x157A + 1*n, n=0...23	Sector 229 SHA384 Hash	See section 6.3.99.232
0x1592 + 1*n, n=0...23	Sector 230 SHA384 Hash	See section 6.3.99.233
0x15AA + 1*n, n=0...23	Sector 231 SHA384 Hash	See section 6.3.99.234
0x15C2 + 1*n, n=0...23	Sector 232 SHA384 Hash	See section 6.3.99.235
0x15DA + 1*n, n=0...23	Sector 233 SHA384 Hash	See section 6.3.99.236
0x15F2 + 1*n, n=0...23	Sector 234 SHA384 Hash	See section 6.3.99.237
0x160A + 1*n, n=0...23	Sector 235 SHA384 Hash	See section 6.3.99.238
0x1622 + 1*n, n=0...23	Sector 236 SHA384 Hash	See section 6.3.99.239
0x163A + 1*n, n=0...23	Sector 237 SHA384 Hash	See section 6.3.99.240
0x1652 + 1*n, n=0...23	Sector 238 SHA384 Hash	See section 6.3.99.241
0x166A + 1*n, n=0...23	Sector 239 SHA384 Hash	See section 6.3.99.242
0x1682 + 1*n, n=0...23	Sector 240 SHA384 Hash	See section 6.3.99.243
0x169A + 1*n, n=0...23	Sector 241 SHA384 Hash	See section 6.3.99.244
0x16B2 + 1*n, n=0...23	Sector 242 SHA384 Hash	See section 6.3.99.245
0x16CA + 1*n, n=0...23	Sector 243 SHA384 Hash	See section 6.3.99.246
0x16E2 + 1*n, n=0...23	Sector 244 SHA384 Hash	See section 6.3.99.247
0x16FA + 1*n, n=0...23	Sector 245 SHA384 Hash	See section 6.3.99.248
0x1712 + 1*n, n=0...23	Sector 246 SHA384 Hash	See section 6.3.99.249
0x172A + 1*n, n=0...23	Sector 247 SHA384 Hash	See section 6.3.99.250
0x1742 + 1*n, n=0...23	Sector 248 SHA384 Hash	See section 6.3.99.251
0x175A + 1*n, n=0...23	Sector 249 SHA384 Hash	See section 6.3.99.252
0x1772 + 1*n, n=0...23	Sector 250 SHA384 Hash	See section 6.3.99.253
0x178A + 1*n, n=0...23	Sector 251 SHA384 Hash	See section 6.3.99.254
0x17A2 + 1*n, n=0...23	Sector 252 SHA384 Hash	See section 6.3.99.255
0x17BA + 1*n, n=0...23	Sector 253 SHA384 Hash	See section 6.3.99.256
0x17D2 + 1*n, n=0...23	Sector 254 SHA384 Hash	See section 6.3.99.257

6.3.99.1 Start Section Pointer - 0x0000

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	Start Section Pointer	0x92	

6.3.99.2 End Section Pointer - 0x0001

Bits	Field Name	Default NVM Value	Description
15	Pointer Type	0x1	Pointer Type: 0b = word units 1b = 4KB sector units Valid values are: 0x0 = Word units 0x1 = 4KB sector units
14:0	End Section Pointer	0xAE	

6.3.99.3 Sector 0 SHA384 Hash[n] (0x0002 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.4 Sector 1 SHA384 Hash[n] (0x001A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.5 Sector 2 SHA384 Hash[n] (0x0032 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.6 Sector 3 SHA384 Hash[n] (0x004A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.7 Sector 4 SHA384 Hash[n] (0x0062 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.8 Sector 5 SHA384 Hash[n] (0x007A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.9 Sector 6 SHA384 Hash[n] (0x0092 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.10 Sector 7 SHA384 Hash[n] (0x00AA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.11 Sector 8 SHA384 Hash[n] (0x00C2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.12 Sector 9 SHA384 Hash[n] (0x00DA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.13 Sector 10 SHA384 Hash[n] (0x00F2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.14 Sector 11 SHA384 Hash[n] (0x010A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.15 Sector 12 SHA384 Hash[n] (0x0122 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.16 Sector 13 SHA384 Hash[n] (0x013A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.17 Sector 14 SHA384 Hash[n] (0x0152 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.18 Sector 15 SHA384 Hash[n] (0x016A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.19 Sector 16 SHA384 Hash[n] (0x0182 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.20 Sector 17 SHA384 Hash[n] (0x019A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.21 Sector 18 SHA384 Hash[n] (0x01B2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.22 Sector 19 SHA384 Hash[n] (0x01CA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.23 Sector 20 SHA384 Hash[n] (0x01E2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.24 Sector 21 SHA384 Hash[n] (0x01FA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.25 Sector 22 SHA384 Hash[n] (0x0212 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.26 Sector 23 SHA384 Hash[n] (0x022A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.27 Sector 24 SHA384 Hash[n] (0x0242 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.28 Sector 25 SHA384 Hash[n] (0x025A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.29 Sector 26 SHA384 Hash[n] (0x0272 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.30 Sector 27 SHA384 Hash[n] (0x028A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.31 Sector 28 SHA384 Hash[n] (0x02A2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.32 Sector 29 SHA384 Hash[n] (0x02BA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.33 Sector 30 SHA384 Hash[n] (0x02D2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.34 Sector 31 SHA384 Hash[n] (0x02EA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.35 Sector 32 SHA384 Hash[n] (0x0302 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.36 Sector 33 SHA384 Hash[n] (0x031A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.37 Sector 34 SHA384 Hash[n] (0x0332 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.38 Sector 35 SHA384 Hash[n] (0x034A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.39 Sector 36 SHA384 Hash[n] (0x0362 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.40 Sector 37 SHA384 Hash[n] (0x037A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.41 Sector 38 SHA384 Hash[n] (0x0392 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.42 Sector 39 SHA384 Hash[n] (0x03AA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.43 Sector 40 SHA384 Hash[n] (0x03C2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.44 Sector 41 SHA384 Hash[n] (0x03DA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.45 Sector 42 SHA384 Hash[n] (0x03F2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.46 Sector 43 SHA384 Hash[n] (0x040A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.47 Sector 44 SHA384 Hash[n] (0x0422 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.48 Sector 45 SHA384 Hash[n] (0x043A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.49 Sector 46 SHA384 Hash[n] (0x0452 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.50 Sector 47 SHA384 Hash[n] (0x046A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.51 Sector 48 SHA384 Hash[n] (0x0482 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.52 Sector 49 SHA384 Hash[n] (0x049A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.53 Sector 50 SHA384 Hash[n] (0x04B2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.54 Sector 51 SHA384 Hash[n] (0x04CA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.55 Sector 52 SHA384 Hash[n] (0x04E2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.56 Sector 53 SHA384 Hash[n] (0x04FA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.57 Sector 54 SHA384 Hash[n] (0x0512 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.58 Sector 55 SHA384 Hash[n] (0x052A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.59 Sector 56 SHA384 Hash[n] (0x0542 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.60 Sector 57 SHA384 Hash[n] (0x055A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.61 Sector 58 SHA384 Hash[n] (0x0572 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.62 Sector 59 SHA384 Hash[n] (0x058A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.63 Sector 60 SHA384 Hash[n] (0x05A2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.64 Sector 61 SHA384 Hash[n] (0x05BA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.65 Sector 62 SHA384 Hash[n] (0x05D2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.66 Sector 63 SHA384 Hash[n] (0x05EA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.67 Sector 64 SHA384 Hash[n] (0x0602 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.68 Sector 65 SHA384 Hash[n] (0x061A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.69 Sector 66 SHA384 Hash[n] (0x0632 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.70 Sector 67 SHA384 Hash[n] (0x064A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.71 Sector 68 SHA384 Hash[n] (0x0662 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.72 Sector 69 SHA384 Hash[n] (0x067A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.73 Sector 70 SHA384 Hash[n] (0x0692 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.74 Sector 71 SHA384 Hash[n] (0x06AA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.75 Sector 72 SHA384 Hash[n] (0x06C2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.76 Sector 73 SHA384 Hash[n] (0x06DA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.77 Sector 74 SHA384 Hash[n] (0x06F2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.78 Sector 75 SHA384 Hash[n] (0x070A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.79 Sector 76 SHA384 Hash[n] (0x0722 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.80 Sector 77 SHA384 Hash[n] (0x073A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.81 Sector 78 SHA384 Hash[n] (0x0752 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.82 Sector 79 SHA384 Hash[n] (0x076A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.83 Sector 80 SHA384 Hash[n] (0x0782 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.84 Sector 81 SHA384 Hash[n] (0x079A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.85 Sector 82 SHA384 Hash[n] (0x07B2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.86 Sector 83 SHA384 Hash[n] (0x07CA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.87 Sector 84 SHA384 Hash[n] (0x07E2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.88 Sector 85 SHA384 Hash[n] (0x07FA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.89 Sector 86 SHA384 Hash[n] (0x0812 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.90 Sector 87 SHA384 Hash[n] (0x082A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.91 Sector 88 SHA384 Hash[n] (0x0842 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.92 Sector 89 SHA384 Hash[n] (0x085A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.93 Sector 90 SHA384 Hash[n] (0x0872 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.94 Sector 91 SHA384 Hash[n] (0x088A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.95 Sector 92 SHA384 Hash[n] (0x08A2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.96 Sector 93 SHA384 Hash[n] (0x08BA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.97 Sector 94 SHA384 Hash[n] (0x08D2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.98 Sector 95 SHA384 Hash[n] (0x08EA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.99 Sector 96 SHA384 Hash[n] (0x0902 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.100 Sector 97 SHA384 Hash[n] (0x091A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.101 Sector 98 SHA384 Hash[n] (0x0932 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.102 Sector 99 SHA384 Hash[n] (0x094A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.103 Sector 100 SHA384 Hash[n] (0x0962 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.104 Sector 101 SHA384 Hash[n] (0x097A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.105 Sector 102 SHA384 Hash[n] (0x0992 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.106 Sector 103 SHA384 Hash[n] (0x09AA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.107 Sector 104 SHA384 Hash[n] (0x09C2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.108 Sector 105 SHA384 Hash[n] (0x09DA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.109 Sector 106 SHA384 Hash[n] (0x09F2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.110 Sector 107 SHA384 Hash[n] (0x0A0A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.111 Sector 108 SHA384 Hash[n] (0x0A22 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.112 Sector 109 SHA384 Hash[n] (0x0A3A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.113 Sector 110 SHA384 Hash[n] (0x0A52 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.114 Sector 111 SHA384 Hash[n] (0x0A6A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.115 Sector 112 SHA384 Hash[n] (0x0A82 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.116 Sector 113 SHA384 Hash[n] (0x0A9A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.117 Sector 114 SHA384 Hash[n] (0x0AB2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.118 Sector 115 SHA384 Hash[n] (0x0ACA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.119 Sector 116 SHA384 Hash[n] (0x0AE2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.120 Sector 117 SHA384 Hash[n] (0x0AFA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.121 Sector 118 SHA384 Hash[n] (0x0B12 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.122 Sector 119 SHA384 Hash[n] (0x0B2A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.123 Sector 120 SHA384 Hash[n] (0x0B42 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.124 Sector 121 SHA384 Hash[n] (0x0B5A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.125 Sector 122 SHA384 Hash[n] (0x0B72 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.126 Sector 123 SHA384 Hash[n] (0x0B8A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.127 Sector 124 SHA384 Hash[n] (0x0BA2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.128 Sector 125 SHA384 Hash[n] (0x0BBA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.129 Sector 126 SHA384 Hash[n] (0x0BD2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.130 Sector 127 SHA384 Hash[n] (0x0BEA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.131 Sector 128 SHA384 Hash[n] (0x0C02 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.132 Sector 129 SHA384 Hash[n] (0x0C1A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.133 Sector 130 SHA384 Hash[n] (0x0C32 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.134 Sector 131 SHA384 Hash[n] (0x0C4A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.135 Sector 132 SHA384 Hash[n] (0x0C62 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.136 Sector 133 SHA384 Hash[n] (0x0C7A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.137 Sector 134 SHA384 Hash[n] (0x0C92 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.138 Sector 135 SHA384 Hash[n] (0x0CAA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.139 Sector 136 SHA384 Hash[n] (0x0CC2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.140 Sector 137 SHA384 Hash[n] (0x0CDA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.141 Sector 138 SHA384 Hash[n] (0x0CF2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.142 Sector 139 SHA384 Hash[n] (0x0D0A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.143 Sector 140 SHA384 Hash[n] (0x0D22 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.144 Sector 141 SHA384 Hash[n] (0x0D3A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.145 Sector 142 SHA384 Hash[n] (0x0D52 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.146 Sector 143 SHA384 Hash[n] (0x0D6A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.147 Sector 144 SHA384 Hash[n] (0x0D82 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.148 Sector 145 SHA384 Hash[n] (0x0D9A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.149 Sector 146 SHA384 Hash[n] (0x0DB2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.150 Sector 147 SHA384 Hash[n] (0x0DCA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.151 Sector 148 SHA384 Hash[n] (0x0DE2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.152 Sector 149 SHA384 Hash[n] (0x0DFA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.153 Sector 150 SHA384 Hash[n] (0x0E12 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.154 Sector 151 SHA384 Hash[n] (0x0E2A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.155 Sector 152 SHA384 Hash[n] (0x0E42 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.156 Sector 153 SHA384 Hash[n] (0x0E5A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.157 Sector 154 SHA384 Hash[n] (0x0E72 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.158 Sector 155 SHA384 Hash[n] (0x0E8A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.159 Sector 156 SHA384 Hash[n] (0x0EA2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.160 Sector 157 SHA384 Hash[n] (0x0EBA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.161 Sector 158 SHA384 Hash[n] (0x0ED2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.162 Sector 159 SHA384 Hash[n] (0x0EEA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.163 Sector 160 SHA384 Hash[n] (0x0F02 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.164 Sector 161 SHA384 Hash[n] (0x0F1A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.165 Sector 162 SHA384 Hash[n] (0x0F32 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.166 Sector 163 SHA384 Hash[n] (0x0F4A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.167 Sector 164 SHA384 Hash[n] (0x0F62 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.168 Sector 165 SHA384 Hash[n] (0x0F7A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.169 Sector 166 SHA384 Hash[n] (0x0F92 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.170 Sector 167 SHA384 Hash[n] (0x0FAA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.171 Sector 168 SHA384 Hash[n] (0x0FC2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.172 Sector 169 SHA384 Hash[n] (0x0FDA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.173 Sector 170 SHA384 Hash[n] (0x0FF2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.174 Sector 171 SHA384 Hash[n] (0x100A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.175 Sector 172 SHA384 Hash[n] (0x1022 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.176 Sector 173 SHA384 Hash[n] (0x103A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.177 Sector 174 SHA384 Hash[n] (0x1052 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.178 Sector 175 SHA384 Hash[n] (0x106A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.179 Sector 176 SHA384 Hash[n] (0x1082 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.180 Sector 177 SHA384 Hash[n] (0x109A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.181 Sector 178 SHA384 Hash[n] (0x10B2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.182 Sector 179 SHA384 Hash[n] (0x10CA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.183 Sector 180 SHA384 Hash[n] (0x10E2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.184 Sector 181 SHA384 Hash[n] (0x10FA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.185 Sector 182 SHA384 Hash[n] (0x1112 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.186 Sector 183 SHA384 Hash[n] (0x112A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.187 Sector 184 SHA384 Hash[n] (0x1142 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.188 Sector 185 SHA384 Hash[n] (0x115A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.189 Sector 186 SHA384 Hash[n] (0x1172 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.190 Sector 187 SHA384 Hash[n] (0x118A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.191 Sector 188 SHA384 Hash[n] (0x11A2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.192 Sector 189 SHA384 Hash[n] (0x11BA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.193 Sector 190 SHA384 Hash[n] (0x11D2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.194 Sector 191 SHA384 Hash[n] (0x11EA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.195 Sector 192 SHA384 Hash[n] (0x1202 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.196 Sector 193 SHA384 Hash[n] (0x121A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.197 Sector 194 SHA384 Hash[n] (0x1232 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.198 Sector 195 SHA384 Hash[n] (0x124A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.199 Sector 196 SHA384 Hash[n] (0x1262 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.200 Sector 197 SHA384 Hash[n] (0x127A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.201 Sector 198 SHA384 Hash[n] (0x1292 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.202 Sector 199 SHA384 Hash[n] (0x12AA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.203 Sector 200 SHA384 Hash[n] (0x12C2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.204 Sector 201 SHA384 Hash[n] (0x12DA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.205 Sector 202 SHA384 Hash[n] (0x12F2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.206 Sector 203 SHA384 Hash[n] (0x130A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.207 Sector 204 SHA384 Hash[n] (0x1322 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.208 Sector 205 SHA384 Hash[n] (0x133A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.209 Sector 206 SHA384 Hash[n] (0x1352 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.210 Sector 207 SHA384 Hash[n] (0x136A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.211 Sector 208 SHA384 Hash[n] (0x1382 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.212 Sector 209 SHA384 Hash[n] (0x139A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.213 Sector 210 SHA384 Hash[n] (0x13B2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.214 Sector 211 SHA384 Hash[n] (0x13CA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.215 Sector 212 SHA384 Hash[n] (0x13E2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.216 Sector 213 SHA384 Hash[n] (0x13FA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.217 Sector 214 SHA384 Hash[n] (0x1412 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.218 Sector 215 SHA384 Hash[n] (0x142A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.219 Sector 216 SHA384 Hash[n] (0x1442 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.220 Sector 217 SHA384 Hash[n] (0x145A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.221 Sector 218 SHA384 Hash[n] (0x1472 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.222 Sector 219 SHA384 Hash[n] (0x148A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.223 Sector 220 SHA384 Hash[n] (0x14A2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.224 Sector 221 SHA384 Hash[n] (0x14BA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.225 Sector 222 SHA384 Hash[n] (0x14D2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.226 Sector 223 SHA384 Hash[n] (0x14EA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.227 Sector 224 SHA384 Hash[n] (0x1502 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.228 Sector 225 SHA384 Hash[n] (0x151A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.229 Sector 226 SHA384 Hash[n] (0x1532 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.230 Sector 227 SHA384 Hash[n] (0x154A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.231 Sector 228 SHA384 Hash[n] (0x1562 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.232 Sector 229 SHA384 Hash[n] (0x157A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.233 Sector 230 SHA384 Hash[n] (0x1592 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.234 Sector 231 SHA384 Hash[n] (0x15AA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.235 Sector 232 SHA384 Hash[n] (0x15C2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.236 Sector 233 SHA384 Hash[n] (0x15DA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.237 Sector 234 SHA384 Hash[n] (0x15F2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.238 Sector 235 SHA384 Hash[n] (0x160A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.239 Sector 236 SHA384 Hash[n] (0x1622 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.240 Sector 237 SHA384 Hash[n] (0x163A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.241 Sector 238 SHA384 Hash[n] (0x1652 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.242 Sector 239 SHA384 Hash[n] (0x166A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.243 Sector 240 SHA384 Hash[n] (0x1682 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.244 Sector 241 SHA384 Hash[n] (0x169A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.245 Sector 242 SHA384 Hash[n] (0x16B2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.246 Sector 243 SHA384 Hash[n] (0x16CA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.247 Sector 244 SHA384 Hash[n] (0x16E2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.248 Sector 245 SHA384 Hash[n] (0x16FA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.249 Sector 246 SHA384 Hash[n] (0x1712 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.250 Sector 247 SHA384 Hash[n] (0x172A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.251 Sector 248 SHA384 Hash[n] (0x1742 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.252 Sector 249 SHA384 Hash[n] (0x175A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.253 Sector 250 SHA384 Hash[n] (0x1772 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.254 Sector 251 SHA384 Hash[n] (0x178A + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.255 Sector 252 SHA384 Hash[n] (0x17A2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.256 Sector 253 SHA384 Hash[n] (0x17BA + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.99.257 Sector 254 SHA384 Hash[n] (0x17D2 + 1*n, n=0...23)

Bits	Field Name	Default NVM Value	Description
15:0	SHA384 Hash	0x0	

6.3.100 NVM Provisioning Area Section Summary Table

Free area used as the new bank when updating 4 MB long modules.

Word Offset	Description	Reference
0x21D000	Raw Data	See section 6.3.100.1

6.3.100.1 Raw Data - 0x21D000

Raw data module length: variable

6.3.101 OROM Section Summary Table

Option ROM Module.

It contains pre-boot code and settings read by BIOS

Word Offset	Description	Reference
0x0000	OROM Data	See section 6.3.101.1

6.3.101.1 OROM Data - 0x0000

Raw data module length: variable

6.3.102 OROM ISK Object CSS Header Section Summary Table

Word Offset	Description	Reference
0x0000	moduleTypeL	See section 6.3.102.1
0x0001	moduleTypeH	See section 6.3.102.2
0x0002	headerLenL	See section 6.3.102.3
0x0003	headerLenH	See section 6.3.102.4
0x0004	headerVersionL	See section 6.3.102.5
0x0005	headerVersionH	See section 6.3.102.6
0x0006	moduleIDL	See section 6.3.102.7
0x0007	moduleIDH	See section 6.3.102.8
0x0008	moduleVendorL	See section 6.3.102.9
0x0009	moduleVendorH	See section 6.3.102.10
0x000A	dateL	See section 6.3.102.11
0x000B	dateH	See section 6.3.102.12
0x000C	sizeL	See section 6.3.102.13
0x000D	sizeH	See section 6.3.102.14
0x000E	keySizeL	See section 6.3.102.15
0x000F	keySizeH	See section 6.3.102.16
0x0010	modulusSizeL	See section 6.3.102.17
0x0011	modulusSizeH	See section 6.3.102.18
0x0012	exponentSizeL	See section 6.3.102.19
0x0013	exponentSizeH	See section 6.3.102.20
0x0014	lad_srevL	See section 6.3.102.21
0x0015	lad_srevH	See section 6.3.102.22
0x0016	crypto_infoL	See section 6.3.102.23
0x0017	crypto_infoH	See section 6.3.102.24
0x0018	lad_fw_entry_offsetL	See section 6.3.102.25
0x0019	lad_fw_entry_offsetH	See section 6.3.102.26
0x001A	reserved2L	See section 6.3.102.27
0x001B	reserved2H	See section 6.3.102.28
0x001C	lad_image_unique_idL	See section 6.3.102.29
0x001D	lad_image_unique_idH	See section 6.3.102.30
0x001E	lad_module_idL	See section 6.3.102.31
0x001F	lad_module_idH	See section 6.3.102.32
0x0020 + 1*n, n=0...31	reserved	See section 6.3.102.33

6.3.102.1 moduleTypeL - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	moduleType	0x6	

6.3.102.2 moduleTypeH - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	moduleTypeH		

6.3.102.3 headerLenL - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	headerLenL	0x1A1	

6.3.102.4 headerLenH - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	headerLenH		

6.3.102.5 headerVersionL - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionL	0x00010000	

6.3.102.6 headerVersionH - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionH		

6.3.102.7 moduleIDL - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	moduleIDL	0x00	

6.3.102.8 moduleIDH - 0x0007

Bits	Field Name	Default NVM Value	Description
15	signMode	0x1	
14:0	moduleIDH		

6.3.102.9 moduleVendorL - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorL	0x00008086	

6.3.102.10 moduleVendorH - 0x0009

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorH		

6.3.102.11 dateL - 0x000A

0xMMDD

Bits	Field Name	Default NVM Value	Description
15:0	DateL	0x20130530	0xMMDD

6.3.102.12 dateH - 0x000B

0xYYYY

Bits	Field Name	Default NVM Value	Description
15:0	DateH		

6.3.102.13 sizeL - 0x000C

Bits	Field Name	Default NVM Value	Description
15:0	sizeL	0x1B6	

6.3.102.14 sizeH - 0x000D

Bits	Field Name	Default NVM Value	Description
15:0	sizeH		

6.3.102.15 keySizeL - 0x000E

Bits	Field Name	Default NVM Value	Description
15:0	keySizeL	0x60	

6.3.102.16 keySizeH - 0x000F

Bits	Field Name	Default NVM Value	Description
15:0	keySizeH		

6.3.102.17 modulusSizeL - 0x0010

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeL	0x60	

6.3.102.18 modulusSizeH - 0x0011

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeH		

6.3.102.19 exponentSizeL - 0x0012

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeL	0x1	

6.3.102.20 exponentSizeH - 0x0013

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeH		

6.3.102.21 lad_srevL - 0x0014

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevL	0x0	

6.3.102.22 lad_srevH - 0x0015

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevH		

6.3.102.23 crypto_infoL - 0x0016

Bits	Field Name	Default NVM Value	Description
15:8	hash algorithm	0x1	
7:0	sign mode	0x8	

6.3.102.24 crypto_infoH - 0x0017

Bits	Field Name	Default NVM Value	Description
15:8	reserved	0x0	
7:0	encryption algorithm	0x0	

6.3.102.25 lad_fw_entry_offsetL - 0x0018

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetL	0x34A	

6.3.102.26 lad_fw_entry_offsetH - 0x0019

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetH		

6.3.102.27 reserved2L - 0x001A

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.102.28 reserved2H - 0x001B

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.102.29 lad_image_unique_idL - 0x001C

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idL	0x0	

6.3.102.30 lad_image_unique_idH - 0x001D

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idH		

6.3.102.31 lad_module_idL - 0x001E

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idL	0xF0	Valid values are: 0x1 = EMP Image 0x2 = PE Image 0x3 = PCIe Analog 0x4 = PHY Analog 0x5 = Option ROM 0xB = Recovery 0xF0 = ISK Object

6.3.102.32 lad_module_idH - 0x001F

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idH		

6.3.102.33 reserved[n] (0x0020 + 1*n, n=0...31)

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.103 OROM ISK Object Key and Signature Section Summary Table

Word Offset	Description	Reference
0x0000 + 1*n, n=0...191	RSA Public Key	See section 6.3.103.1
0x00C0 + 1*n, n=0...191	RSA Public Key Prime	See section 6.3.103.2
0x0180 + 1*n, n=0...191	RSA Public Key Second	See section 6.3.103.3
0x0240	RSA ExponentL	See section 6.3.103.4
0x0241	RSA ExponentH	See section 6.3.103.5
0x0242 + 1*n, n=0...191	Encrypted SHA384 Hash	See section 6.3.103.6

6.3.103.1 RSA Public Key[n] (0x0000 + 1*n, n=0...191)

Associated with the public portion of the Key Authentication Key used to validate this ISK.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.3.103.2 RSA Public Key Prime[n] (0x00C0 + 1*n, n=0...191)

Associated with the public portion of the Key Authentication Key used to validate this ISK.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key Prime		

6.3.103.3 RSA Public Key Second[n] (0x0180 + 1*n, n=0...191)

Associated with the public portion of the Key Authentication Key used to validate this ISK.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key Second		

6.3.103.4 RSA ExponentL - 0x0240

Associated with the public portion of the Key Authentication Key used to validate this ISK.

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentL	0x0	

6.3.103.5 RSA ExponentH - 0x0241

Associated with the public portion of the Key Authentication Key used to validate this ISK.

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentH	0x0	

6.3.103.6 Encrypted SHA384 Hash[n] (0x0242 + 1*n, n=0...191)

Bits	Field Name	Default NVM Value	Description
15:0	Encrypted SHA384 Hash	0x0	

6.3.104 OROM ISK Object Auth Header Section Summary Table

Word Offset	Description	Reference
0x0000	Device Blank NVM Device ID	See section 6.3.104.1
0x0001	Max Module AreaL	See section 6.3.104.2
0x0002	Max Module AreaH	See section 6.3.104.3
0x0003	Current Module AreaL	See section 6.3.104.4
0x0004	Current Module AreaH	See section 6.3.104.5
0x0005	Reserved	See section 6.3.104.6
0x0006	Code Revision	See section 6.3.104.7
0x0007	Reserved Spare Word	See section 6.3.104.8

6.3.104.1 Device Blank NVM Device ID - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Device Blank NVM Device ID	0x57AC	

6.3.104.2 Max Module AreaL - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaL	0x36C	

6.3.104.3 Max Module AreaH - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaH	0x0	

6.3.104.4 Current Module AreaL - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaL	0x36C	

6.3.104.5 Current Module AreaH - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaH	0x0000	

6.3.104.6 Reserved - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	

6.3.104.7 Code Revision - 0x0006

Bits	Field Name	Default NVM Value	Description
15:8	Major Revision	0x0	
7:0	Minor Revision	0x0	

6.3.104.8 Reserved Spare Word - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	Reserved Spare Word	0x0	

6.3.105 OROM ISK Object Section Summary Table

Word Offset	Description	Reference
0x0000	ISK ID	See section 6.3.105.1
0x0001 + 1*n, n=0...31	ISK SHA512 Hash	See section 6.3.105.2

6.3.105.1 ISK ID - 0x0000

Bits	Field Name	Default NVM Value	Description
15	Revocation Scheme	0x1	0x0 - Pseudo timer based revocation 0x1 - Explicit revocation
14:0	Key Revocation ID	0x0	Indicates the index of the fuse used to revoke this ISK.

6.3.105.2 ISK SHA512 Hash[n] (0x0001 + 1*n, n=0...31)

Bits	Field Name	Default NVM Value	Description
15:0	ISK SHA512 Hash	0x0	

6.3.106 OROM CSS Header Section Summary Table

Word Offset	Description	Reference
0x0000	moduleTypeL	See section 6.3.106.1
0x0001	moduleTypeH	See section 6.3.106.2
0x0002	headerLenL	See section 6.3.106.3
0x0003	headerLenH	See section 6.3.106.4
0x0004	headerVersionL	See section 6.3.106.5
0x0005	headerVersionH	See section 6.3.106.6
0x0006	moduleIDL	See section 6.3.106.7
0x0007	moduleIDH	See section 6.3.106.8
0x0008	moduleVendorL	See section 6.3.106.9
0x0009	moduleVendorH	See section 6.3.106.10
0x000A	dateL	See section 6.3.106.11
0x000B	dateH	See section 6.3.106.12
0x000C	sizeL	See section 6.3.106.13
0x000D	sizeH	See section 6.3.106.14
0x000E	keySizeL	See section 6.3.106.15
0x000F	keySizeH	See section 6.3.106.16
0x0010	modulusSizeL	See section 6.3.106.17
0x0011	modulusSizeH	See section 6.3.106.18
0x0012	exponentSizeL	See section 6.3.106.19
0x0013	exponentSizeH	See section 6.3.106.20
0x0014	lad_srevL	See section 6.3.106.21
0x0015	lad_srevH	See section 6.3.106.22
0x0016	crypto_infoL	See section 6.3.106.23
0x0017	crypto_infoH	See section 6.3.106.24
0x0018	lad_fw_entry_offsetL	See section 6.3.106.25
0x0019	lad_fw_entry_offsetH	See section 6.3.106.26
0x001A	reserved2L	See section 6.3.106.27
0x001B	reserved2H	See section 6.3.106.28
0x001C	lad_image_unique_idL	See section 6.3.106.29
0x001D	lad_image_unique_idH	See section 6.3.106.30
0x001E	lad_module_idL	See section 6.3.106.31
0x001F	lad_module_idH	See section 6.3.106.32
0x0020	reserved	See section 6.3.106.33

6.3.106.1 moduleTypeL - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	moduleType	0x6	

6.3.106.2 moduleTypeH - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	moduleTypeH		

6.3.106.3 headerLenL - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	headerLenL	0xa1	

6.3.106.4 headerLenH - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	headerLenH		

6.3.106.5 headerVersionL - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionL	0x00010000	

6.3.106.6 headerVersionH - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionH		

6.3.106.7 moduleIDL - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	moduleIDL	0x0	

6.3.106.8 moduleIDH - 0x0007

Bits	Field Name	Default NVM Value	Description
15	signMode	0x1	
14:0	moduleIDH		

6.3.106.9 moduleVendorL - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorL	0x00008086	

6.3.106.10 moduleVendorH - 0x0009

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorH		

6.3.106.11 dateL - 0x000A

0xMMDD

Bits	Field Name	Default NVM Value	Description
15:0	DateL	0x20130530	0xMMDD

6.3.106.12 dateH - 0x000B

0xYYYY

Bits	Field Name	Default NVM Value	Description
15:0	DateH		

6.3.106.13 sizeL - 0x000C

size is in DWords

Bits	Field Name	Default NVM Value	Description
15:0	sizeL	0x00025800	

6.3.106.14 sizeH - 0x000D

size is in DWords

Bits	Field Name	Default NVM Value	Description
15:0	sizeH		

6.3.106.15 keySizeL - 0x000E

Bits	Field Name	Default NVM Value	Description
15:0	keySizeL	0x40	

6.3.106.16 keySizeH - 0x000F

Bits	Field Name	Default NVM Value	Description
15:0	keySizeH		

6.3.106.17 modulusSizeL - 0x0010

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeL	0x40	

6.3.106.18 modulusSizeH - 0x0011

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeH		

6.3.106.19 exponentSizeL - 0x0012

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeL	0x1	

6.3.106.20 exponentSizeH - 0x0013

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeH		

6.3.106.21 lad_srevL - 0x0014

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevL	0x1	

6.3.106.22 lad_srevH - 0x0015

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevH		

6.3.106.23 crypto_infoL - 0x0016

Bits	Field Name	Default NVM Value	Description
15:8	hash algorithm	0x1	
7:0	sign mode	0x8	

6.3.106.24 crypto_infoH - 0x0017

Bits	Field Name	Default NVM Value	Description
15:12	reserved	0x0	
11:8	srev_flags	0x2	
7:0	encryption algorithm	0x0	

6.3.106.25 lad_fw_entry_offsetL - 0x0018

offset of OROM image from the beginning of OROM CSS module

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetL	0x14c	

6.3.106.26 lad_fw_entry_offsetH - 0x0019

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetH		

6.3.106.27 reserved2L - 0x001A

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.106.28 reserved2H - 0x001B

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.106.29 lad_image_unique_idL - 0x001C

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idL		

6.3.106.30 lad_image_unique_idH - 0x001D

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idH		

6.3.106.31 lad_module_idL - 0x001E

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idL	0x5	Valid values are: 0x1 = EMP Image 0x2 = PE Image 0x3 = PCIe Analog 0x4 = PHY Analog 0x5 = Option ROM

6.3.106.32 lad_module_idH - 0x001F

Bits	Field Name	Default NVM Value	Description
15	Reserved	0x0	
14:0	lad_module_idH		

6.3.106.33 reserved - 0x0020

Raw data module length: variable

6.3.107 OROM Key and Signature Section Summary Table

Word Offset	Description	Reference
0x0000 + 1*n, n=0...191	RSA Public Key	See section 6.3.107.1
0x00C0 + 1*n, n=0...191	RSA Public Key Prime	See section 6.3.107.2
0x0180 + 1*n, n=0...191	RSA Public Key Second	See section 6.3.107.3
0x0240	RSA ExponentL	See section 6.3.107.4
0x0241	RSA ExponentH	See section 6.3.107.5
0x0242 + 1*n, n=0...191	Encrypted SHA384 Hash	See section 6.3.107.6

6.3.107.1 RSA Public Key[n] (0x0000 + 1*n, n=0...191)

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.3.107.2 RSA Public Key Prime[n] (0x00C0 + 1*n, n=0...191)

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key Prime	0x0	

6.3.107.3 RSA Public Key Second[n] (0x0180 + 1*n, n=0...191)

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key Second	0x0	

6.3.107.4 RSA ExponentL - 0x0240

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentL	0x0	

6.3.107.5 RSA ExponentH - 0x0241

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentH	0x0	

6.3.107.6 Encrypted SHA384 Hash[n] (0x0242 + 1*n, n=0...191)

Bits	Field Name	Default NVM Value	Description
15:0	Encrypted SHA384 Hash	0x0	

6.3.108 OROM Auth Header Section Summary Table

Word Offset	Description	Reference
0x0000	Device Blank NVM Device ID	See section 6.3.108.1
0x0001	Max Module AreaL	See section 6.3.108.2
0x0002	Max Module AreaH	See section 6.3.108.3
0x0003	Current Module AreaL	See section 6.3.108.4
0x0004	Current Module AreaH	See section 6.3.108.5
0x0005	Reserved	See section 6.3.108.6
0x0006	Code Revision	See section 6.3.108.7
0x0007	Reserved Spare Word	See section 6.3.108.8

6.3.108.1 Device Blank NVM Device ID - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Device Blank NVM Device ID	0x57AC	

6.3.108.2 Max Module AreaL - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaL	0x1000	

6.3.108.3 Max Module AreaH - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaH	0x0009	

6.3.108.4 Current Module AreaL - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaL	0x1000	

6.3.108.5 Current Module AreaH - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaH	0x0000	

6.3.108.6 Reserved - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	

6.3.108.7 Code Revision - 0x0006

Bits	Field Name	Default NVM Value	Description
15:8	Major Revision	0x0	
7:0	Minor Revision	0x0	

6.3.108.8 Reserved Spare Word - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	Reserved Spare Word	0x0	

6.3.109 OROM Provisioning Area Section Summary Table

500 KB OROM free area used as the new bank.

Word Offset	Description	Reference
0x0000	Raw Data	See section 6.3.109.1

6.3.109.1 Raw Data - 0x0000

Raw data module length: variable

6.3.110 Link Topology Netlist Section Summary Table

MD Link Topology

Word Offset	Description	Reference
0x0000	Link Topology Netlist Header	See section 6.3.110.1
0x0001	Length	See section 6.3.110.2
0x0002	Link Topology Netlist Data	See section 6.3.110.3

6.3.110.1 Link Topology Netlist Header - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0x11B	Valid values are: 0x11B = MD Link Topology (0x11B)

6.3.110.2 Length - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Length		Length in: 2 Bytes unit - 2 First Section -> Word: Link Topology Netlist -> Link Topology Netlist Header Last Section -> Word: Link Topology Netlist -> Link Topology Netlist Data

6.3.110.3 Link Topology Netlist Data - 0x0002

Raw data module length: 10236 words

Link Topology Netlist deliverable content

6.3.111 Link Topology Netlist ISK Object CSS Header Section Summary Table

Word Offset	Description	Reference
0x0000	moduleTypeL	See section 6.3.111.1
0x0001	moduleTypeH	See section 6.3.111.2
0x0002	headerLenL	See section 6.3.111.3
0x0003	headerLenH	See section 6.3.111.4
0x0004	headerVersionL	See section 6.3.111.5
0x0005	headerVersionH	See section 6.3.111.6
0x0006	moduleIDL	See section 6.3.111.7
0x0007	moduleIDH	See section 6.3.111.8
0x0008	moduleVendorL	See section 6.3.111.9
0x0009	moduleVendorH	See section 6.3.111.10
0x000A	dateL	See section 6.3.111.11
0x000B	dateH	See section 6.3.111.12
0x000C	sizeL	See section 6.3.111.13
0x000D	sizeH	See section 6.3.111.14
0x000E	keySizeL	See section 6.3.111.15
0x000F	keySizeH	See section 6.3.111.16
0x0010	modulusSizeL	See section 6.3.111.17
0x0011	modulusSizeH	See section 6.3.111.18
0x0012	exponentSizeL	See section 6.3.111.19
0x0013	exponentSizeH	See section 6.3.111.20
0x0014	lad_srevL	See section 6.3.111.21
0x0015	lad_srevH	See section 6.3.111.22
0x0016	crypto_infoL	See section 6.3.111.23
0x0017	crypto_infoH	See section 6.3.111.24
0x0018	lad_fw_entry_offsetL	See section 6.3.111.25
0x0019	lad_fw_entry_offsetH	See section 6.3.111.26
0x001A	reserved2L	See section 6.3.111.27
0x001B	reserved2H	See section 6.3.111.28
0x001C	lad_image_unique_idL	See section 6.3.111.29
0x001D	lad_image_unique_idH	See section 6.3.111.30
0x001E	lad_module_idL	See section 6.3.111.31
0x001F	lad_module_idH	See section 6.3.111.32
0x0020 + 1*n, n=0...31	reserved	See section 6.3.111.33

6.3.111.1 moduleTypeL - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	moduleType	0x6	

6.3.111.2 moduleTypeH - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	moduleTypeH		

6.3.111.3 headerLenL - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	headerLenL	0x1A1	

6.3.111.4 headerLenH - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	headerLenH		

6.3.111.5 headerVersionL - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionL	0x00010000	

6.3.111.6 headerVersionH - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionH		

6.3.111.7 moduleIDL - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	moduleIDL	0x00	

6.3.111.8 moduleIDH - 0x0007

Bits	Field Name	Default NVM Value	Description
15	signMode	0x1	
14:0	moduleIDH		

6.3.111.9 moduleVendorL - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorL	0x00008086	

6.3.111.10 moduleVendorH - 0x0009

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorH		

6.3.111.11 dateL - 0x000A

0xMMDD

Bits	Field Name	Default NVM Value	Description
15:0	DateL	0x20130530	0xMMDD

6.3.111.12 dateH - 0x000B

0xYYYY

Bits	Field Name	Default NVM Value	Description
15:0	DateH		

6.3.111.13 sizeL - 0x000C

Bits	Field Name	Default NVM Value	Description
15:0	sizeL	0x1B6	

6.3.111.14 sizeH - 0x000D

Bits	Field Name	Default NVM Value	Description
15:0	sizeH		

6.3.111.15 keySizeL - 0x000E

Bits	Field Name	Default NVM Value	Description
15:0	keySizeL	0x60	

6.3.111.16 keySizeH - 0x000F

Bits	Field Name	Default NVM Value	Description
15:0	keySizeH		

6.3.111.17 modulusSizeL - 0x0010

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeL	0x60	

6.3.111.18 modulusSizeH - 0x0011

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeH		

6.3.111.19 exponentSizeL - 0x0012

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeL	0x1	

6.3.111.20 exponentSizeH - 0x0013

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeH		

6.3.111.21 lad_srevL - 0x0014

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevL	0x0	

6.3.111.22 lad_srevH - 0x0015

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevH		

6.3.111.23 crypto_infoL - 0x0016

Bits	Field Name	Default NVM Value	Description
15:8	hash algorithm	0x1	
7:0	sign mode	0x8	

6.3.111.24 crypto_infoH - 0x0017

Bits	Field Name	Default NVM Value	Description
15:8	reserved	0x0	
7:0	encryption algorithm	0x0	

6.3.111.25 lad_fw_entry_offsetL - 0x0018

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetL	0x34A	

6.3.111.26 lad_fw_entry_offsetH - 0x0019

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetH		

6.3.111.27 reserved2L - 0x001A

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.111.28 reserved2H - 0x001B

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.111.29 lad_image_unique_idL - 0x001C

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idL	0x0	

6.3.111.30 lad_image_unique_idH - 0x001D

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idH		

6.3.111.31 lad_module_idL - 0x001E

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idL	0xF0	Valid values are: 0x1 = EMP Image 0x2 = PE Image 0x3 = PCIe Analog 0x4 = PHY Analog 0x5 = Option ROM 0xB = Recovery 0xF0 = ISK Object

6.3.111.32 lad_module_idH - 0x001F

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idH		

6.3.111.33 reserved[n] (0x0020 + 1*n, n=0...31)

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.112 Link Topology Netlist ISK Object Key and Signature Section Summary Table

Word Offset	Description	Reference
0x0000 + 1*n, n=0...191	RSA Public Key	See section 6.3.112.1
0x00C0 + 1*n, n=0...191	RSA Public Key Prime	See section 6.3.112.2
0x0180 + 1*n, n=0...191	RSA Public Key Second	See section 6.3.112.3
0x0240	RSA ExponentL	See section 6.3.112.4
0x0241	RSA ExponentH	See section 6.3.112.5
0x0242 + 1*n, n=0...191	Encrypted SHA384 Hash	See section 6.3.112.6

6.3.112.1 RSA Public Key[n] (0x0000 + 1*n, n=0...191)

Associated with the public portion of the Key Authentication Key used to validate this ISK.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.3.112.2 RSA Public Key Prime[n] (0x00C0 + 1*n, n=0...191)

Associated with the public portion of the Key Authentication Key used to validate this ISK.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key Prime		

6.3.112.3 RSA Public Key Second[n] (0x0180 + 1*n, n=0...191)

Associated with the public portion of the Key Authentication Key used to validate this ISK.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key Second		

6.3.112.4 RSA ExponentL - 0x0240

Associated with the public portion of the Key Authentication Key used to validate this ISK.

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentL	0x0	

6.3.112.5 RSA ExponentH - 0x0241

Associated with the public portion of the Key Authentication Key used to validate this ISK.

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentH	0x0	

6.3.112.6 Encrypted SHA384 Hash[n] (0x0242 + 1*n, n=0...191)

Bits	Field Name	Default NVM Value	Description
15:0	Encrypted SHA384 Hash	0x0	

6.3.113 Link Topology Netlist ISK Object Auth Header Section Summary Table

Word Offset	Description	Reference
0x0000	Device Blank NVM Device ID	See section 6.3.113.1
0x0001	Max Module AreaL	See section 6.3.113.2
0x0002	Max Module AreaH	See section 6.3.113.3
0x0003	Current Module AreaL	See section 6.3.113.4
0x0004	Current Module AreaH	See section 6.3.113.5
0x0005	Reserved	See section 6.3.113.6
0x0006	Code Revision	See section 6.3.113.7
0x0007	Reserved Spare Word	See section 6.3.113.8

6.3.113.1 Device Blank NVM Device ID - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Device Blank NVM Device ID	0x57AC	

6.3.113.2 Max Module AreaL - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaL	0x36C	

6.3.113.3 Max Module AreaH - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaH	0x0	

6.3.113.4 Current Module AreaL - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaL	0x36C	

6.3.113.5 Current Module AreaH - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaH	0x0000	

6.3.113.6 Reserved - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	

6.3.113.7 Code Revision - 0x0006

Bits	Field Name	Default NVM Value	Description
15:8	Major Revision	0x0	
7:0	Minor Revision	0x0	

6.3.113.8 Reserved Spare Word - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	Reserved Spare Word	0x0	

6.3.114 Link Topology Netlist ISK Object Section Summary Table

Word Offset	Description	Reference
0x0000	ISK ID	See section 6.3.114.1
0x0001 + 1*n, n=0...31	ISK SHA512 Hash	See section 6.3.114.2

6.3.114.1 ISK ID - 0x0000

Bits	Field Name	Default NVM Value	Description
15	Revocation Scheme	0x1	0x0 - Pseudo timer based revocation 0x1 - Explicit revocation
14:0	Key Revocation ID	0x0	Indicates the index of the fuse used to revoke this ISK.

6.3.114.2 ISK SHA512 Hash[n] (0x0001 + 1*n, n=0...31)

Bits	Field Name	Default NVM Value	Description
15:0	ISK SHA512 Hash	0x0	

6.3.115 Link Topology Netlist CSS Header Section Summary Table

Word Offset	Description	Reference
0x0000	moduleTypeL	See section 6.3.115.1
0x0001	moduleTypeH	See section 6.3.115.2
0x0002	headerLenL	See section 6.3.115.3
0x0003	headerLenH	See section 6.3.115.4
0x0004	headerVersionL	See section 6.3.115.5
0x0005	headerVersionH	See section 6.3.115.6
0x0006	moduleIDL	See section 6.3.115.7
0x0007	moduleIDH	See section 6.3.115.8
0x0008	moduleVendorL	See section 6.3.115.9
0x0009	moduleVendorH	See section 6.3.115.10
0x000A	dateL	See section 6.3.115.11
0x000B	dateH	See section 6.3.115.12
0x000C	sizeL	See section 6.3.115.13
0x000D	sizeH	See section 6.3.115.14
0x000E	keySizeL	See section 6.3.115.15
0x000F	keySizeH	See section 6.3.115.16
0x0010	modulusSizeL	See section 6.3.115.17
0x0011	modulusSizeH	See section 6.3.115.18
0x0012	exponentSizeL	See section 6.3.115.19
0x0013	exponentSizeH	See section 6.3.115.20
0x0014	lad_srevL	See section 6.3.115.21
0x0015	lad_srevH	See section 6.3.115.22
0x0016	crypto_infoL	See section 6.3.115.23
0x0017	crypto_infoH	See section 6.3.115.24
0x0018	lad_fw_entry_offsetL	See section 6.3.115.25
0x0019	lad_fw_entry_offsetH	See section 6.3.115.26
0x001A	reserved2L	See section 6.3.115.27
0x001B	reserved2H	See section 6.3.115.28
0x001C	lad_image_unique_idL	See section 6.3.115.29
0x001D	lad_image_unique_idH	See section 6.3.115.30
0x001E	lad_module_idL	See section 6.3.115.31
0x001F	lad_module_idH	See section 6.3.115.32
0x0020 + 1*n, n=0...31	reserved	See section 6.3.115.33

6.3.115.1 moduleTypeL - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	moduleType	0x6	

6.3.115.2 moduleTypeH - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	moduleTypeH		

6.3.115.3 headerLenL - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	headerLenL	0x1a1	

6.3.115.4 headerLenH - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	headerLenH		

6.3.115.5 headerVersionL - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionL	0x00010000	

6.3.115.6 headerVersionH - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	headerVersionH		

6.3.115.7 moduleIDL - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	moduleIDL	0x00	

6.3.115.8 moduleIDH - 0x0007

Bits	Field Name	Default NVM Value	Description
15	signMode	0x1	
14:0	moduleIDH		

6.3.115.9 moduleVendorL - 0x0008

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorL	0x00008086	

6.3.115.10 moduleVendorH - 0x0009

Bits	Field Name	Default NVM Value	Description
15:0	moduleVendorH		

6.3.115.11 dateL - 0x000A

0xMMDD

Bits	Field Name	Default NVM Value	Description
15:0	DateL	0x20130530	0xMMDD

6.3.115.12 dateH - 0x000B

0xYYYY

Bits	Field Name	Default NVM Value	Description
15:0	DateH		

6.3.115.13 sizeL - 0x000C

Bits	Field Name	Default NVM Value	Description
15:0	sizeL	0x0000FA00	

6.3.115.14 sizeH - 0x000D

Bits	Field Name	Default NVM Value	Description
15:0	sizeH		

6.3.115.15 keySizeL - 0x000E

Bits	Field Name	Default NVM Value	Description
15:0	keySizeL	0x60	

6.3.115.16 keySizeH - 0x000F

Bits	Field Name	Default NVM Value	Description
15:0	keySizeH		

6.3.115.17 modulusSizeL - 0x0010

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeL	0x60	

6.3.115.18 modulusSizeH - 0x0011

Bits	Field Name	Default NVM Value	Description
15:0	modulusSizeH		

6.3.115.19 exponentSizeL - 0x0012

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeL	0x1	

6.3.115.20 exponentSizeH - 0x0013

Bits	Field Name	Default NVM Value	Description
15:0	exponentSizeH		

6.3.115.21 lad_srevL - 0x0014

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevL	0x1	

6.3.115.22 lad_srevH - 0x0015

Bits	Field Name	Default NVM Value	Description
15:0	lad_srevH		

6.3.115.23 crypto_infoL - 0x0016

Bits	Field Name	Default NVM Value	Description
15:8	hash algorithm	0x1	
7:0	sign mode	0x8	

6.3.115.24 crypto_infoH - 0x0017

Bits	Field Name	Default NVM Value	Description
15:12	reserved	0x0	
11:8	srev_flags	0x1	
7:0	encryption algorithm	0x0	

6.3.115.25 lad_fw_entry_offsetL - 0x0018

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetL	0x6C0	

6.3.115.26 lad_fw_entry_offsetH - 0x0019

Bits	Field Name	Default NVM Value	Description
15:0	lad_fw_entry_offsetH		

6.3.115.27 reserved2L - 0x001A

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.115.28 reserved2H - 0x001B

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.115.29 lad_image_unique_idL - 0x001C

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idL	0x0	

6.3.115.30 lad_image_unique_idH - 0x001D

Bits	Field Name	Default NVM Value	Description
15:0	lad_image_unique_idH		

6.3.115.31 lad_module_idL - 0x001E

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idL	0xB	Valid values are: 0x1 = EMP Image 0x2 = PE Image 0x3 = PCIe Analog 0x4 = PHY Analog 0x5 = Option ROM 0xB = Recovery

6.3.115.32 lad_module_idH - 0x001F

Bits	Field Name	Default NVM Value	Description
15:0	lad_module_idH		

6.3.115.33 reserved[n] (0x0020 + 1*n, n=0...31)

Bits	Field Name	Default NVM Value	Description
15:0	reserved	0x0	

6.3.116 Link Topology Netlist Key and Signature Section Summary Table

Word Offset	Description	Reference
0x0000 + 1*n, n=0...191	RSA Public Key	See section 6.3.116.1
0x00C0 + 1*n, n=0...191	RSA Public Key Prime	See section 6.3.116.2
0x0180 + 1*n, n=0...191	RSA Public Key Second	See section 6.3.116.3
0x0240	RSA ExponentL	See section 6.3.116.4
0x0241	RSA ExponentH	See section 6.3.116.5
0x0242 + 1*n, n=0...191	Encrypted SHA384 Hash	See section 6.3.116.6

6.3.116.1 RSA Public Key[n] (0x0000 + 1*n, n=0...191)

Associated with the public portion of the Image Signing Key used to validate this image/

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key	0x0	

6.3.116.2 RSA Public Key Prime[n] (0x00C0 + 1*n, n=0...191)

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key Prime		

6.3.116.3 RSA Public Key Second[n] (0x0180 + 1*n, n=0...191)

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA Public Key Second		

6.3.116.4 RSA ExponentL - 0x0240

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentL	0x0	

6.3.116.5 RSA ExponentH - 0x0241

Associated with the public portion of the Image Signing Key used to validate this image.

Bits	Field Name	Default NVM Value	Description
15:0	RSA ExponentH	0x0	

6.3.116.6 Encrypted SHA384 Hash[n] (0x0242 + 1*n, n=0...191)

Bits	Field Name	Default NVM Value	Description
15:0	Encrypted SHA384 Hash	0x0	

6.3.117 Link Topology Netlist Auth Header Section Summary Table

Word Offset	Description	Reference
0x0000	Device Blank NVM Device ID	See section 6.3.117.1
0x0001	Max Module AreaL	See section 6.3.117.2
0x0002	Max Module AreaH	See section 6.3.117.3
0x0003	Current Module AreaL	See section 6.3.117.4
0x0004	Current Module AreaH	See section 6.3.117.5
0x0005	Reserved	See section 6.3.117.6
0x0006	Code Revision	See section 6.3.117.7
0x0007	Reserved Spare Word	See section 6.3.117.8

6.3.117.1 Device Blank NVM Device ID - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Device Blank NVM Device ID	0x57AC	

6.3.117.2 Max Module AreaL - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaL	0x1000	

6.3.117.3 Max Module AreaH - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	Max Module AreaH	0x0009	

6.3.117.4 Current Module AreaL - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaL	0xE000	

6.3.117.5 Current Module AreaH - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	Current Module AreaH	0x0006	

6.3.117.6 Reserved - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	Reserved	0xFFFF	

6.3.117.7 Code Revision - 0x0006

Bits	Field Name	Default NVM Value	Description
15:8	Major Revision	0x0	
7:0	Minor Revision	0x0	

6.3.117.8 Reserved Spare Word - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	Reserved Spare Word	0x0	

6.3.118 TLV extension Provisioning Area Section Summary Table

Word Offset	Description	Reference
0x0000	Sub Module Type padding	See section 6.3.118.1
0x0001	Length	See section 6.3.118.2

6.3.118.1 Sub Module Type padding - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Sub Module Type	0xffff	Valid values are: 0xffff = Padding Module

6.3.118.2 Length - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Length	0x0	Length in: 2 Bytes unit - 2 First Section -> Word: TLV extension Provisioning Area -> Sub Module Type padding Last Section -> Word: TLV extension Provisioning Area -> Length

6.3.119 Link Topology Scratch Pad Area Section Summary Table

Link topology scratch pad area used by Firmware only.

Word Offset	Description	Reference
0x0000	[New Word]	See section 6.3.119.1

6.3.119.1 [New Word] - 0x0000

Raw data module length: variable

6.3.120 FW Scratch Pad Area Section Summary Table

FW scratch pad area used by FW only.

Word Offset	Description	Reference
0x0000	[New Word]	See section 6.3.120.1

6.3.120.1 [New Word] - 0x0000

Raw data module length: variable

6.3.121 Factory Settings Header Section Summary Table

Word Offset	Description	Reference
0x0000	actual size_L	See section 6.3.121.1
0x0001	actual size H	See section 6.3.121.2
0x0002	password	See section 6.3.121.3
0x0003	TLV Extension Offset	See section 6.3.121.4
0x0004	TLV Extension Size	See section 6.3.121.5
0x0005	PCIR AL Offset	See section 6.3.121.6
0x0006	POR AL Offset	See section 6.3.121.7
0x0007	PCI Serial id MAC Address Offset	See section 6.3.121.8
0x0008	Reserved	See section 6.3.121.9

6.3.121.1 actual size_L - 0x0000

Bits	Field Name	Default NVM Value	Description
15:0	Actual Size_L	0xFFFF	

6.3.121.2 actual size H - 0x0001

Bits	Field Name	Default NVM Value	Description
15:0	Actual Size H	0xFFFF	

6.3.121.3 password - 0x0002

Bits	Field Name	Default NVM Value	Description
15:0	password	0xFFFF	

6.3.121.4 TLV Extension Offset - 0x0003

Bits	Field Name	Default NVM Value	Description
15:0	TLV Extension offset	0xFFFF	

6.3.121.5 TLV Extension Size - 0x0004

Bits	Field Name	Default NVM Value	Description
15:0	TLV Extension Size	0xFFFF	

6.3.121.6 PCIR AL Offset - 0x0005

Bits	Field Name	Default NVM Value	Description
15:0	PCIR AL Offset	0xFFFF	

6.3.121.7 POR AL Offset - 0x0006

Bits	Field Name	Default NVM Value	Description
15:0	POR AL Offset	0xFFFF	

6.3.121.8 PCI Serial id MAC Address Offset - 0x0007

Bits	Field Name	Default NVM Value	Description
15:0	PCI Serial id MAC Address Section	0xFFFF	

6.3.121.9 Reserved - 0x0008

Raw data module length: variable

6.3.122 Factory Settings Area Section Summary Table

Word Offset	Description	Reference
0x0000	Reserved	See section 6.3.122.1

6.3.122.1 Reserved - 0x0000

Raw data module length: variable

6.3.123 Guarded Zone Section Summary Table

Guarded Zone for FW use.

The Zone is protected such that only FW can read and write to it.

Word Offset	Description	Reference
0x0000	[New Word]	See section 6.3.123.1

6.3.123.1 [New Word] - 0x0000

Raw data module length: variable



NOTE: *This page intentionally left blank.*

Chapter 7 Inline Functions

7.1 Receive Functionality

Packet reception consists of:

- Recognizing the presence of a packet on the wire ([Section 7.1.1](#))
- Performing address filtering ([Section 7.1.2](#))
- Checksum off-loads ([Section 7.1.6](#))
- DMA queue assignment ([Section 7.1.3](#))
- Storing the packet in the receive data FIFO ([Section 7.6.3.1](#))
- Transferring the data to assigned receive queues in host memory ([Section 7.1.4](#))
- Updating the state of a receive descriptor ([Section 7.1.5](#)).

7.1.1 MAC Layer - Receive

7.1.1.1 Packet Acceptance Criteria

In addition to the filtering rules described in the next sections, a packet must also meet the following criteria to be accepted by the device:

1. Normally, only good packets are received (packets with none of the following errors: Under Size Error, Over Size Error, Packet Error, Length Error and CRC Error). However, if the *Store Bad Packets* bit is set (FCTRL.SBP), bad packets that don't pass the filter function are stored in host memory. Packet errors are indicated by error bits in the receive descriptor (RDESC.ERRORS). It is possible to receive all packets, regardless of whether they are bad, by setting the promiscuous enables bit and the *Store Bad Packets* bit. In this case, bad packets are queued according to the same rules as regular packets.
2. Min. Packet Size (Runt packets) — Rx packets, smaller than 48 bytes, cannot be posted to host memory regardless of save bad frame setting.
3. Max Packet Size — Any Rx packet posted from the MAC unit to the DMA unit cannot exceed 15.5 KB. Further restrictions per queue are described in [Section 7.1.4](#).
4. CRC errors before the Start Frame Delimiter (SFD) are ignored. All packets must have a valid SFD to be recognized by the device (even bad packets).

7.1.1.2 CRC Strip

The E610 potentially strips the L2 CRC on incoming packets.

CRC strip is enabled by the HLREG0.RXCRCSTRP bit. When set, CRC is stripped from all received packets.

The global CRC strip bit (HLREG0.RXCRCSTRP) must be set in the following cases were the packet changes before being handled to the driver:

- VLAN is hidden (PFQDE.HIDE_VLAN = 1) in any queue.
- E-tag is removed in any queue (PFQDE.STRIP_TAG is set)
- L2 tags are stripped from packets (PFQDE.STRIP_TAG = 1) in any queue.
- Time stamp is added to the packets in any TC (any bit of TSYNCRXCTL.TSIP_UT_EN or TSYNCRXCTL.TSIP_UP_EN is set).
- Short received packets are padded (RDRXCTL.PSP is set).

7.1.2 Packet Filtering

A received packet goes through up to three stages of filtering as depicted in [Figure 7-1](#). The figure describes a switch-like structure that is used in virtualization mode to route packets between the network port (top of drawing) and one of many virtual ports (bottom of drawing), where each virtual port might be associated with a Virtual Machine (VM), a Virtual Machine Monitor (VMM), or the like. The three stages are:

1. **First stage — Admission Control:** Ensure that the packet should be received by the port. This is done by a set of L2 filters and is described in detail in this section.
 2. **Second stage — Pooling:** This stage is specific to virtualization environments and defines the virtual ports (called pools in this document) that are the targets for the Rx packet. A packet can be associated with any number of ports/pools and the selection process is described in [Section 7.1.3.2](#). In non virtualization mode, this stage is skipped and all the queues used in the next stage are considered as part of the same default pool.
- Note:** A pool is equivalent to a VSI as defined in IEEE 802.1Qbg specification.
3. **Third stage — Queueing:** A receive packet that successfully passed the Rx filters is associated with one of many receive descriptor queues as described in [Section 7.1.3](#).

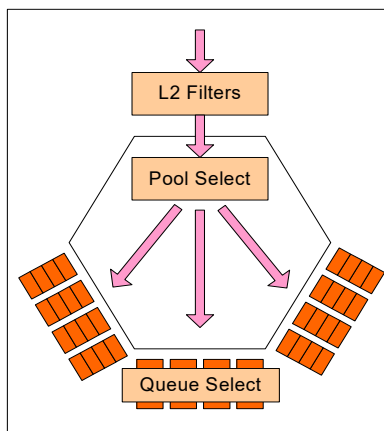


Figure 7-1. Stages in Packet Filtering - Virtualization Mode

The receive packet filtering role is to determine which of the incoming packets are allowed to pass to the local machine and which of the incoming packets should be dropped since they are not targeted to the local machine. Received packets that are targeted for the local machine can be destined to the host, to a manageability controller, or to both. This section describes how host filtering is done, and the interaction with management filtering.

As depicted in [Figure 7-2](#), host filtering is done in two stages:

1. Packets are filtered by L2 filters (Ethernet MAC Address, unicast/multicast/broadcast). See [Section 7.1.2.1](#).
2. Packets are filtered by VLAN if a VLAN tag is present. See [Section 7.1.2.2](#).

A packet is not forwarded to the host if any of the following occurs:

- The packet does not pass L2 filters, as described in [Section 7.1.2.1](#).
- The packet does not pass VLAN filtering, as described in [Section 7.1.2.2](#).
- The packet passes manageability filtering and the manageability filters determine that the packet should not pass to the host (see MNGONLY register and [Section 10.4.5.2](#)).

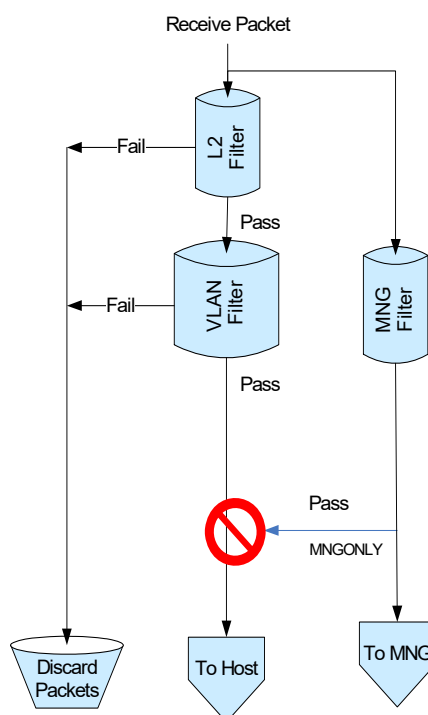


Figure 7-2. Rx Filtering Flow Chart

7.1.2.1 L2 Filtering

A packet passes successfully through L2 Ethernet MAC Address filtering if any of the following conditions are met:

- **Unicast packet filtering** — Promiscuous unicast filtering is enabled ($FCTRL.UPE = 1b$) or the packet passes unicast MAC filters (host).
- **Multicast packet filtering** — Promiscuous multicast filtering is enabled ($FCTRL.MPE = 1b$) or the packet matches one of the multicast filters.
- **Broadcast packet filtering to host** — Promiscuous multicast filtering is enabled ($FCTRL.MPE = 1b$) or Broadcast Accept Mode is enabled ($FCTRL.BAM = 1b$).

7.1.2.1.1 Unicast Filter

The Ethernet MAC Address is checked against the 128 host unicast addresses and 4 KB hash-based unicast address filters (if enabled). The host unicast addresses are controlled by the host interface. The destination address of an incoming packet must exactly match one of the pre-configured host address filters. These addresses can be unicast or multicast. Those filters are configured through Receive Address Low (RAL), Receive Address High (RAH), In addition, there are 4 KB unicast hash filters used for host defined by the PFUTA registers. The unicast hash filters are useful mainly for virtualization settings in those cases that more than 128 filters might be required.

Promiscuous Unicast — Receive all unicasts. Promiscuous unicast mode is usually used when the LAN device is used as a sniffer.

7.1.2.1.2 Multicast Filter (Partial)

The 12-bit portion of the incoming packet multicast address must be set in the multicast filter address table (*MTA*) to pass the partial multicast filter. The bits (out of 48 bits of the destination address) used to index the MTA table can be selected by the *MO* field in the MCSTCTRL register.

Promiscuous Multicast — Receive all multicasts. Promiscuous multicast mode is usually used when the LAN device is used as a sniffer.

7.1.2.2 VLAN Filtering

The E610 provides exact VLAN filtering as follows:

- Host VLAN filters are programmed by the VFTA[n] registers.
- A VLAN match might relate to the *DEI* bit in the VLAN header. It is enabled for host filtering only by the VLNCTRL.DEIEN while the expected value is defined by the VLNCTRL.DEI.

If double VLAN is enabled (see [Section 7.4.5](#)), filtering is done on the second (internal) VLAN tag. All the filtering functions of the E610 ignore the first (external) VLAN in this mode.

A receive packet that passes MAC Address filtering successfully is subjected to VLAN header filtering as illustrated in [Figure 7-3](#):

1. If the packet does not have a VLAN header, it passes to the next filtering stage.
2. Else, if VLAN filters are not enabled (VLNCTRL.VFE = 0b), the packet is forwarded to the next filtering stage.
3. Else, if the packet matches an enabled VLAN filter and DEI checking (if enabled), the packet is forwarded to the next filtering stage.
4. Otherwise, the packet is dropped.

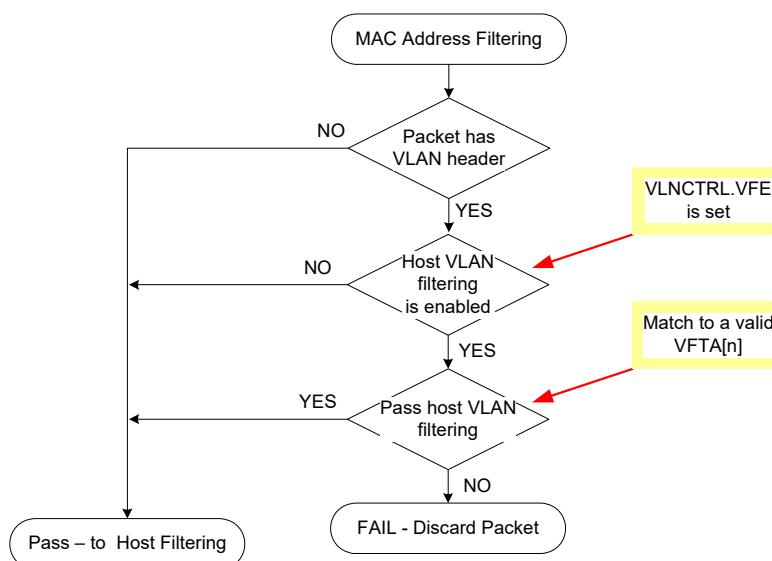


Figure 7-3. VLAN Filtering

7.1.2.3 E-tag Filtering

If the PFVTCTL.POOLING_MODE is E-tag (01b), special L2 filtering rules are applied to tagged packets.

- If the tag matches one of the RAH registers used for tag filtering (RAH.ADTYPE = 1), it is considered as a packet that passed L2.
- If the FCTRL.TPE bit is set (*Tag Promiscuous Enable*), all the tagged packets pass L2 filtering.
- Otherwise, packets with E-tag (if PFVTCTL.POOLING_MODE is E-tag (01b)) are dropped.

Note: E-tag packets are not expected when double VLAN are use, thus PFVTCTL.POOLING_MODE should be cleared if CTRL_EXT.EXTENDED_VLAN is set.

7.1.2.4 Manageability/Host Filtering

The host and manageability filtering process are mostly independent. Each entity defines which packet it should receive. The only exception is that the manageability filters can define a packet as exclusive and thus prevent it from reaching the host. See [Section 10.4.5.2](#) for details. [Figure 7-4](#) describes this flow.

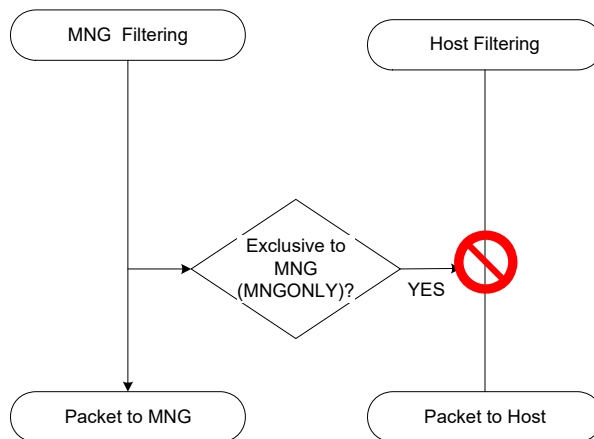


Figure 7-4. Manageability/Host Filtering

7.1.3 Rx Queues Assignment

The following filters/mechanisms determine the destination of a received packet. These filters are described briefly while more detailed descriptions are provided in the following sections:

- **Virtualization** — In a virtual environment, DMA resources are shared between more than one software entity (operating system and/or device driver). This is done by allocating receive descriptor queues to virtual partitions (VMM, IOVM, VMs, or VFs). Allocating queues to virtual partitions is done in sets, each with the same number of queues, called queue pools, or pools. Virtualization assigns to each received packet one or more pool indices. Packets are routed to a pool based on their pool index and other considerations such as DCB and RSS. See [Section 7.1.3.2](#) for more on routing for virtualization.
- **DCB** — Data Center Bridging (DCB) provides QoS through priority queues, priority flow control, and congestion management. Packets are classified into one of several (up to eight) Traffic Classes (TCs). Each TC is associated with a single unique packet buffer. Packets that reside in a specific packet buffer are then routed to one of a set of Rx queues based on their TC value and other considerations such as RSS and virtualization. See [Section 7.6](#) for details on DCB.
 - DCB is enabled via the *DCB_ENA* bit
- **Receive Side Scaling (RSS)** — RSS distributes packet processing between several processor cores by assigning packets into different descriptor queues. RSS assigns to each received packet an RSS index. Packets are routed to one queue from a set of Rx queues based on their RSS index and other considerations such as DCB and virtualization. See [Section 7.1.3.6](#) for details.
- **L2 EtherType Filters** — These filters identify packets by their L2 EtherType and assigns them to receive queues. Examples of possible uses are LLDP packets, and 802.1X packets. See [Section 7.1.3.3](#) for details. The E610 incorporates eight EtherType filters.
- **Flow Director Filters** — These filters that provide up to additional 32 K filters. See [Section 7.1.3.5](#) for details.

- **TCP SYN Filters** — The E610 might route TCP packets with their SYN flag set into a separate queue. SYN packets are often used in SYN attacks to load the system with numerous requests for new connections. By filtering such packets to a separate queue, security software can monitor and act on SYN attacks. See [Section 7.1.3.4](#) for details.

A received packet is allocated to a queue based on the above criteria and the following order:

- Queue by L2 EtherType filters (if match)
- Queue by SYN filter (if match)
- Queue by flow director filters
- Queue (in case of virtualization, within a pool) by DCB and/or RSS as described in [Section 7.1.3.1](#) and [Section 7.1.3.2](#).
- Send to queue zero.

7.1.3.1 Queuing in a Non-virtualized Environment

[Table 7-1](#) lists the queuing schemes for packets that do not match any special filters (L2 EtherType, SYN and flow director filters). [Table 7-2](#) illustrates the queue indexing. Selecting a scheme is done via the *Multiple Receive Queues Enable (MRQE)* field in MRQC register.

Table 7-1. Rx Queuing Schemes Supported (No Virtualization)

DCB	RSS	DCB/RSS Queues	Special Filters ¹
No	No	1 queue Rx queue 0	Supported
No	Yes	64 RSS queues	Supported
Yes	No	8 TCs x 1 queue 4 TCs x 1 queue Assign Rx queue 0 of each TC	Supported
Yes	Yes	8 TCs x 16 RSS 4 TCs x 32 RSS	Supported

1. Special filters include: L2 filters, SYN filter. When possible it is recommended to assign Rx queues not used by the DCB / RSS queues.

Table 7-2. Queue Indexing Illustration in Non-virtualization Mode

MRQC.MRQE	Queue Index Bits	6	5	4	3	2	1	0	
0000b	Default only	0							
0001b	RSS		RSS						
0101b	DCB(4) + RSS	TC		RSS ¹					
0011b	DCB(4)	TC		0					
0100b	DCB(8) + RSS	TC			RSS ¹				
0010b	DCB(8)	TC			0				

1. The number of bits used for each TC is set according to the RQTC.RQTCx fields

A received packet is assigned to a queue according to the ordering shown in [Figure 7-5](#):

- **DCB and RSS filters** — Packets that do not meet any of the previous filtering conditions described in [Section 7.1.3](#) are assigned to one of 128 queues as listed in [Table 7-1](#). The following modes are supported:
 - **No DCB, No RSS** — Queue 0 is used for all packets.
 - **RSS only** — A set of 64 queues is allocated for RSS. The queue is identified through the RSS index. Note that it is possible to use a subset of these queues.
 - **DCB only** — A single queue is allocated per TC to a total of eight queues (if the number of TCs is eight), or to a total of four queues (if the number of TCs is four). The queue is identified through the TC index.
 - **DCB with RSS** — A packet is assigned to one of 128 queues (8 TCs x 16 RSS or 4 TCs x 32 RSS) through the DCB traffic class of the packet and the RSS index. The TC index is used as the MS bits of the Rx queue index, and the LS bits are defined by the RSS index. The number of queues used for each TC are set according to the `RQTC.RQTCx` fields

When operating in conjunction with DCB, the number of RSS queues can vary per DCB TC. Each TC can be configured to a different number of RSS queues (1/2/4/8/16/32 queues). The output of the RSS redirection table is masked accordingly to generate an RSS index of the right width. When configured to less than the maximum number of queues, the respective MS bits of the RSS index are set to zero. The number of RSS queues per TC is configured in the RQTC register.

- **Example** — Assume a 4 TCs x 32 RSS configuration and that the number of RSS queues for TC=3 is set to 4. The queue numbers for TC=3 are 64, 65, 66, and 67 (decimal).

[Figure 7-5](#) depicts the flow of allocation of Rx queues by the various queue filters previously described:

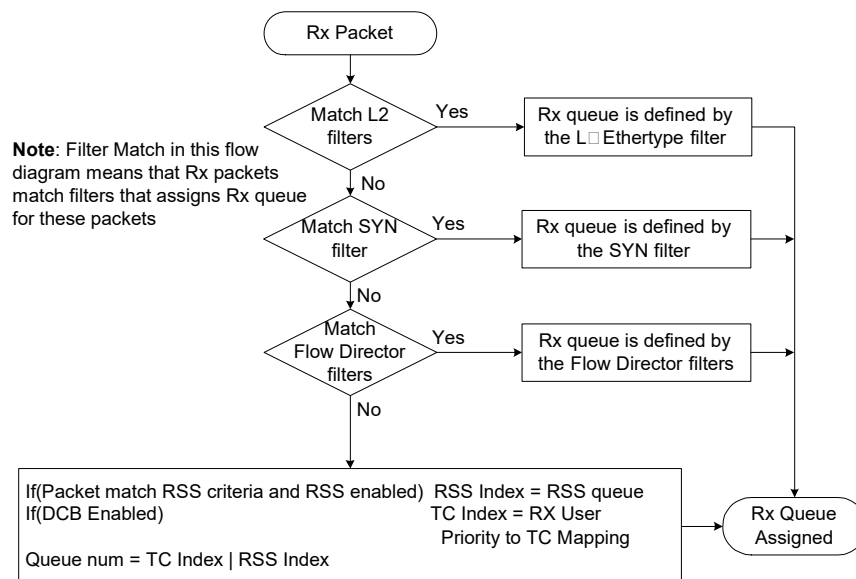


Figure 7-5. Rx Queuing Flow (Non-virtualized)

7.1.3.2 Queuing in a Virtualized Environment

The 128 Rx queues are allocated to a pre-configured number of queue sets, called pools. In non-I/OV mode, system software allocates the pools to the VMM or to VMs. In IOV mode, each pool may be associated with a VF.

Incoming packets are associated with pools based on their L2 characteristics as described in [Section 7.8.3](#). This section describes the following stage, where an Rx queue is assigned to each replication of the Rx packet as determined by its pools association.

[Table 7-3](#) lists the queuing schemes supported with virtualization. [Table 7-4](#) lists the queue indexing.

Table 7-3. Rx Queuing Schemes Supported with Virtualization

DCB	RSS	DCB/RSS Queues	Special Filters ¹
No	No	16 pools x 1 queue 32 pools x 1 queue 64 pools x 1 queue - Rx queue 0 of each pool	Supported
No	Yes	32 pools x 4 RSS 64 pools x 2 RSS	Supported
Yes	No	16 pools x 8 TCs 32 pools x 4 TCs	Supported
Yes	Yes	N/A	N/A

1. Special filters include: L2 filters, SYN filter. When possible it is recommended to assign Rx queues not used by the DCB / RSS queues.

Table 7-4. Queue Indexing Illustration in Virtualization Mode

MRQC.MRQE	Queue Index Bits	6	5	4	3	2	1	0
1000b		Pool Index						0
1011b	VT(64) + RSS	Pool Index						RSS
1001b	VT(64) + RSS with 1 x Mega Pool	Pool Index = 0...61 (000000b...111101b)						RSS
		Pool Index / 2 = 62 / 2 = 11111b					RSS	
1111b	VT(64) + RSS with 2 x Mega Pool	Pool Index = 0...59 = 000000b...111011b						RSS
		Pool Index / 2 = (60... 62) / 2 = 11110b... 11111b					RSS	
1010b	VT(32) + RSS	Pool Index					RSS	
N/A	VT(16) + RSS	Not Supported						
1101b	VT(32) + DCB(4)	Pool Index					TC	
1100b	VT(16) + DCB(8)	Pool Index				TC		
1110b	VT(16) + DCB(4) + RSS	Pool Index				TC		RSS

Selecting a scheme is done in the following manner:

- Non-IOV mode
 - Select one of the above schemes via the *Multiple Receive Queues Enable* field in the MRQC register.
- IOV mode
 - Determine the number of pools: the number must support the value configured by the operating system in the PCIe *NumVFs* field (see [Section 9.2.4.4.5](#)). Therefore, the number of pools is min. of {16, 32, 64} that is still $\geq$ NumVFs.
 - Determine DCB mode via the *DCB_ENA* bit in MTQC register.

A received packet is assigned to an absolute queue index according to the ordering shown in [Figure 7-6](#)). It is software responsibility to define a queue that belongs to the matched pool:

- DCB and RSS filters — The supported modes are listed in [Table 7-3](#) and detailed as follows. The associated queue indexes are listed in [Table 7-4](#).
 - **No DCB, No RSS** — A single queue is used as default queue of the pool with either 16, 32, or 64 pools enabled. In a 64 pools setting, queues '2xN'...'2xN+1' are allocated to pool 'N'; In a 32 pools setting, queues '4xN'...'4xN+3' are allocated to pool 'N'. The queues not used as default may be directed to by other filters.
 - **RSS only** — All 128 queues are allocated to pools. Configurations supported:
 - 32 pools with 4 RSS queues each
 - 64 pools with 2 RSS queues each.
 - 63 pools. The first 62 with 2 queues each and the last one with 4 RSS queues. In this mode the last pool is 62 and pool 63 is disabled.
 - 62 pools. The first 60 with 2 queues each and the last two with 4 RSS queues each. In this mode the last two pools are 60 and 62 and pools 61/63 are disabled.
 - **Note:** It is possible to use a subset of the RSS queues in each pool. The LS bits of the queue indexes are defined by the RSS index, and the pool index is used as the MS bits. See description below.
 - **Note:** In the 62 and 63 pools modes, the pools with 4 queues should be assigned to the PF and can not be assigned to a VF.
 - **DCB only** — All 128 queues are allocated to pools. Two configurations are supported: 16 pools with 8 TCs each or 32 pools with 4 TCs each. The LS bits of the queue indexes are defined by the TC index, and the pool index is used as the MS bits.
 - **DCB and RSS** — All 128 queues are allocated to pools. One configuration is supported: 16 pools with 4 TCs each TC with 2 RSS queues. The LS bit of the queue indexes is defined by the RSS index, the next 2 bits of the queue indexes are defined by the TC index, and the pool index is used as the MS bits.

When operating in conjunction with RSS, the number of RSS queues can vary per pool as defined by the $PSRTYPE[n].RQPL$. Each pool can be configured to a different number of RSS queues (1/2/4- queues) up to the maximum possible queues in the selected mode of operation. The output of the RSS redirection table is masked accordingly to generate an RSS index of the right width. When configured to less than the maximum number of queues, the respective MS bits of the RSS index are set to zero.

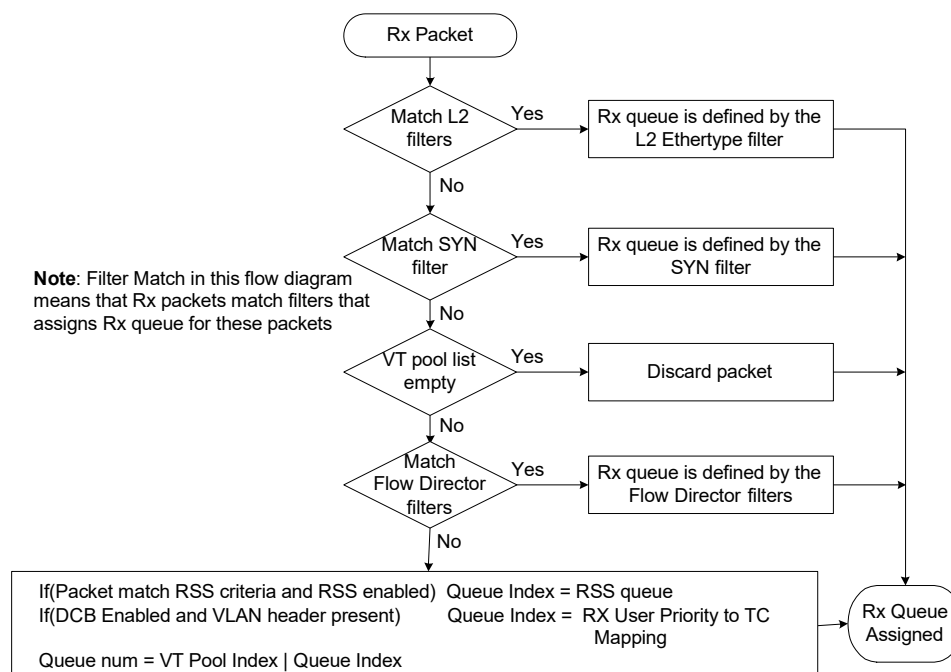


Figure 7-6. Rx Queuing Flow (Virtualization Case)

7.1.3.3 L2 EtherType Filters

These filters identify packets by their L2 EtherType, 802.1Q user priority and optionally assign them to a receive queue. The following possible usages have been identified at this time:

- DCB LLDP packets — Identifies DCB control packets
- IEEE 802.1X packets — Extensible Authentication Protocol (EAPOL) over LAN
- Time sync packets (such as IEEE 1588) — Identifies Sync or Delay_Req packets
- The L2 type filters should not be set to IP packet type as this might cause unexpected results

The E610 incorporates eight EtherType filters defined by a set of two registers per filter: ETQF[n] and ETQS[n].

The L2 packet type is defined by comparing the *EtherType* field in the Rx packet with the ETQF[n].*ETYPE* (regardless of the pool and UP matching). The *Packet Type* field in the Rx descriptor captures the filter number that matched with the L2 EtherType. See [Section 7.1.5.2.2](#) for a description of the *Packet Type* field.

The following flow is used by the EtherType filters:

1. If the ETQF.*FILTER_ENABLE* bit is cleared, the filter is disabled and the following steps are ignored.
2. Receive packet matches any ETQF filters if the *EtherType* field in the packet matches the *ETYPE* field of the filter. Note that the following steps are ignored if the packet does not match the ETQF filters.

Note: EtherType Filters should not be configured in a way that may cause a packet to match multiple filters.

3. Packets that match any ETQF filters are candidate for the host. If the packet also matches the manageability filters, it is directed to the host as well regardless of the MNGONLY register setting.
4. If the ETQF.IEEE_1588_TIME_STAMP field is set, the packet is identified as an IEEE 1588 packet.
5. If the ETQS.QUEUE_ENABLE bit is cleared, the filter completed its action on the packet. Else, the filter is also used for queuing purposes as described in the sections that follow.
6. If the ETQF.POOL_ENABLE field is set, the Pool field of the filter determines the target pool for the packet. The packet can still be mirrored to other pools as described in [Section 7.8.3](#). See the sections that follow for more details on the use of the Pool field.
7. The ETQS.RX_QUEUE field determines the destination queue for the packet. In case of a mirrored packet, only the copy of the packet that is targeted to the pool defined by the Pool field in the ETQF register is routed according to the Rx Queue field.

Setting the ETQF[n] registers is described as follows:

- The FILTER_ENABLE bit enables identification of Rx packets by EtherType according to this filter. If this bit is cleared, the filter is ignored.
- The ETYPE field contains the 16-bit EtherType compared against all L2 type fields in the Rx packet.
- The IEEE_1588_TIME_STAMP bit indicates that the EtherType defined in the EType field is identified as IEEE 1588 EType. Packets that match this filter are time stamped according to the IEEE 1588 specification.
- The Pool field defines the target pool for a packet that matches the filter.
 - It applies only in virtualization modes. The pool index is meaningful only if the POOL_ENABLE bit is set.
 - If the POOL_ENABLE bit is set, the QUEUE_ENABLE bit in the ETQS register must be set as well. In this case, the RX_QUEUE field in the ETQS must be part of the pool number defined in the ETQF.

Setting the ETQS[n] registers is described as follows:

- The QUEUE_ENABLE bit enables routing of the Rx packet that match the filter to Rx queue as defined by the RX_QUEUE field.
- The RX_QUEUE field contains the destination queue (one of 128 queues) for the packet.

Special considerations for virtualization modes:

- Packets that match an EtherType filter are diverted from their original pool (as defined by the VLAN and Ethernet MAC Address filters) to the pool defined in the Pool field in the ETQF registers.
- The same applies for multicast packets. A single copy is posted to the pool defined by the filter.
- Mirroring rules
 - A packet sent to a pool by an ETQF filter, is still candidate to mirroring using the standard mirroring rules.
 - The EtherType filter does not take part in the decision on the destination of the mirrored packet.

7.1.3.4 SYN Packet Filters

The E610 might route TCP packets whose SYN flag is set into a separate queue. SYN packets are used in SYN attacks to load the system with numerous requests for new connections. By filtering such packets to a separate queue, security software can monitor and act on SYN attacks.

The following rules apply:

- A single SYN filter is provided.

The SYN filter is configured via the SYNQF register as follows:

- The *QUEUE_ENABLE* bit enables SYN filtering capability.
- The *RX_QUEUE* field contains the destination queue for the packet (one of 128 queues). In case of mirroring (in virtualization mode), only the original copy of the packet is routed according to this filter.

7.1.3.5 Flow Director Filters

The flow director filters identify specific flows or sets of flows and routes them to specific queues. The flow director filters are programmed by FDIRCTRL and all other FDIR registers. The E610 shares the Rx packet buffer for the storage of these filters.

The E610 supports three different modes of flow director (set in the FDIRCTRL.FILTERMODE field - [Section 8.2.2.14.1](#)):

- IP filtering (000b)
- MAC, VLAN filtering (001b)
- Cloud: NVGRE or VXLAN filtering (010b)

[Table 7-5](#) describes the packets that are candidate to compared to the flow director filters according to the different modes:

Table 7-5. Candidate for Flow Director per Mode

Mode	IP packets TCP/UDP/SCTP (Not Tunneled)	Non IP Packets	NVGRE Packets	VXLAN packets	Other IP packets or Fragmented IP
00 - IP (L4 fields unmasked)	Candidate	Not Candidate	Candidate (inner IP) ¹	Candidate (inner IP) ¹	Not Candidate
00 - IP (L4 fields masked)	Candidate	Not Candidate	Candidate (inner IP) ²	Candidate (inner IP) ²	Candidate ^{2/3}
01 - MAC VLAN	Candidate	Candidate	Candidate	Candidate	Candidate
10 - Cloud	Not Candidate	Not Candidate	Candidate	Candidate	Not Candidate

1. Only if a known L4 header is present (TCP/UDP/SCTP).

2. VXLAN and NVGRE packets without inner IP header (e.g. tunneled ARP) are not candidate in IP mode.

3. IP in IP packets are not candidate in IP mode.

Basic rules for the flow director filters are:

In VT mode, the *Pool* field in FDIRCMD must be valid. If the packet is replicated, only the copy that goes to the pool that matches the *Pool* field is impacted by the filter. The flow director filters cover the following fields:

Table 7-6. Lookup Fields for Flow Director per Mode

Mode	Lookup Fields
00 - IP	- VLAN header - Source IP and destination IP Addresses - Source port and destination port numbers - IPv4 / IPv6¹ and UDP / TCP or SCTP protocol match - Flexible 2-byte tuple anywhere in the first 64 bytes of the packet - Target pool number (relevant only for VT mode)
01 - MAC VLAN	- MAC - VLAN - Flexible 2-byte tuple anywhere in the first 64 bytes of the packet - Target pool number (relevant only for VT mode)
10 - Cloud	- Tunnel type - NVGRE or VXLAN - TNI (NVGRE) / VNI (VXLAN) - Inner MAC - Inner VLAN - Flexible 2-byte tuple anywhere in the first 64 bytes of the packet - Target pool number (relevant only for VT mode)

1. IPv6 extended headers are parsed by the E610, enabling TCP layer header recognition. Still the IPv6 extended header fields are not taken into account for the queue classification by Flow Director filter. This rule do not apply for security headers and fragmentation header. Packets with fragmentation header miss this filter. Packets with security extended headers are parsed only up to these headers and therefore can match only filters that do not require fields from the L4 protocol.

The E610 supports two types of filtering modes (static setting by the `FDIRCTRL.PERFECT_MATCH` bit):

- **Perfect Match Filters** — The hardware checks a match between the masked fields of the received packets and the programmed filters. Masked fields should be programmed as zeros in the filter context. The E610 supports up to 8 K - 1 perfect match filters.
- **Signature Filters** — The hardware checks a match between a hash-based signature of the masked fields of the received packet. The E610 supports up to 32 K - 1 signature filters. This mode can be used only when the filtering mode is IPMODE (`FDIRCTRL.FILTERMODE` = 000b).

Note: The *Perfect Match* fields and *Signature* field are denoted as *Flow ID* fields.

The E610 supports masking / range for the previously described fields. These masks are defined globally for all filters in the `FDIR...M` registers.

- The following fields can be masked per bit enabling power of two ranges up to complete enable / disable of the fields: IPv4 addresses and L4 port numbers.
- The following fields can be masked per byte enabling lower granularity ranges up to complete enable / disable of the fields: IPv6 addresses; Inner MAC; Outer MAC; *TNI/VNI* field.

Note: In perfect match filters the destination IPv6 address can only be compared as a whole (with no range support) to the *IP6AT* filters. A match to any of the *IP6AT* filter is considered as an IPv6 destination match.

- The following fields can be either enabled or disabled completely for the match functionality: VLAN ID tag; Outer VLAN Priority + *DEI* bit (the user priority is taken from the outermost tag with *UP* bits if an inner VLAN exists; otherwise it is ignored. Inner or outer VLAN tag; Flexible 2-byte tuple and target pool. Target pool can be enabled by software only when VT is enabled as well.

Flow director filters have the following functionality in virtualization mode:

- Flow director filters are programmed by the registers in the PF described in [Section 7.1.3.5.13](#) and [Section 7.1.3.5.14](#).
- Flow director filters can be utilized in virtualization mode to filter on MAC-VLAN by setting `FDIRCTRL.FILTERMODE = MACVLANMODE` (001b). `MACVLANMODE` is only used in flow director perfect match mode.

7.1.3.5.1 Flow Director Filters Actions

Flow director filters might have one of the following actions programmed per filter in the `FDIRCMD` register:

- Drop packet or pass to host as defined by the *Drop* bit.
 - Matched packets to a flow director filter is directed to the assigned Rx queue only if the packet does not match the L2 filters for queue assignment nor the SYN filter for queue assignment.
 - Packets that match a filter are directed to the Rx queue defined in the filter context as programmed by the `FDIRCMD.RX_QUEUE`. The `RX_QUEUE` field is an absolute receive queue index. In a non-VT setting, it can be programmed to any value. In VT mode, the software should set the `RX_QUEUE` to an index that belongs to the matched pool.
 - Packets that match drop filters are directed to the Rx queue defined per all filters in the `FDIRCTRL.DROP_QUEUE`. The E610 drops these packets if software does not enable the specific Rx queue.

7.1.3.5.2 Flow Director Default Action

A default drop action may be applied by the flow director for packets that are candidate to flow director as defined in [Table 7-5](#). If the `FDIRCTRL.DROP_NO_MATCH` bit is set, any packet candidate for flow director that does not match any of the filters is sent to the queue defined in the `FDIRCTRL.DROP_QUEUE` field.

7.1.3.5.3 Flow Director Filters Status Reporting

Shared status indications for all packets:

- The E610 increments the `FDIRMATCH` counter for packets that match a flow director filter. It also increments the `FDIRMISS` counter for packets that do not match any flow director filter.
- The *Flow Director Filter Match (FLM)* bit in the *Extended Status* field of the Rx descriptor is set for packets that match a flow director filter.
- The flow ID parameters are reported in the *Flow Director Filter ID* field in the Rx descriptor if enabled by the `FDIRCTRL.REPORT_STATUS`. When the *Report-Status* bit is set, the `RXCSUM.PCSD` bit should be set as well. This field is indicated for all packets that match or do not match the flow director filters.
 - For packets that do not match a flow director filter, if the `FDIRCTRL.REPORT_STATUS_ALWAYS` is set, the *Flow Director Filter ID* field can be used by software for future programming of a matched filter, otherwise, the RSS hash value is reported. [Table 7-7](#) describes the value of the `RSS_TYPE` field in the receive descriptor in the different cases.

Table 7-7. RSS Type Values According to Flow Director Match

<i>REPORT_STATUS</i> (FDIRCTRL[5])	<i>REPORT_STATUS_ALWAYS</i> (FDIRCTRL[7])	Packet matches a Flow Director Entry	RSS Type Field Value
0	0	X	RSS Type
1	0	0	RSS Type
1	0	1	0xF
0	1	X	illegal configuration
1	1	X	0xF

For packets that match a flow director filter, the *Flow Director Filter ID* field can be used by software to identify the flow of the Rx packet. Too long linked list exception (linked list and too long terms are illustrated in [Figure 7-7](#)):

- The maximum recommended linked list length is programmed in the *FDIRCTRL.MAX_LENGTH* field
- The length exception is reported in the *FDIRERR* field in the Rx descriptor
- Packets that do not match any flow director filter, reports this exception if the length of the existing linked list is above the maximum recommended length. Software can use it to avoid further programming of additional filters to this linked list before other filters are removed.
- Packets that match a pass filter report this exception if the distance of the matched filter from the beginning of the linked list is higher than the above recommended length.
- Packets that match a drop filter are posted to the Rx queue programmed in the filter context instead of the global *FDIRCTRL.DROP_QUEUE*. The drop exception is reported in addition to the length exception (in the same field in the Rx descriptor).

Collision exception:

- Packets that matches a collided filter report this exception in the *FDIRERR* field in the Rx descriptor.
- Collision events for signature-based filters should be rare. Still it might happen because multiple flows can have the same hash and signature values. Software might leave the setting as is while the collided flows are handled according to the actions of the first programmed flow. Only one flow (out of the collided ones) might remain in the flow director filters. To clear the collision indication in the programmed filter, software should remove the filter and then re-program it once again.
- Collision events for a perfect match filter should never happen. A collision error might indicate a programming fault that software might decide to fix.

7.1.3.5.4 Flow Director Filters Block Diagram

Figure 7-7 shows a block diagram of the flow director filters. Received flows are identified to buckets by a hash function on the relevant tuples as defined by the FDIR...M registers. Each bucket is organized in a linked list indicated by the hash lookup table. Buckets can have a variable length while the last filter in each bucket is indicated as a last. There is no upper limit for a linked list length during programming. However, a received packet that matches a filter that exceeds the FDIRCTRL.MAX_LENGTH are reported to software (see Section 7.1.3.5.6).

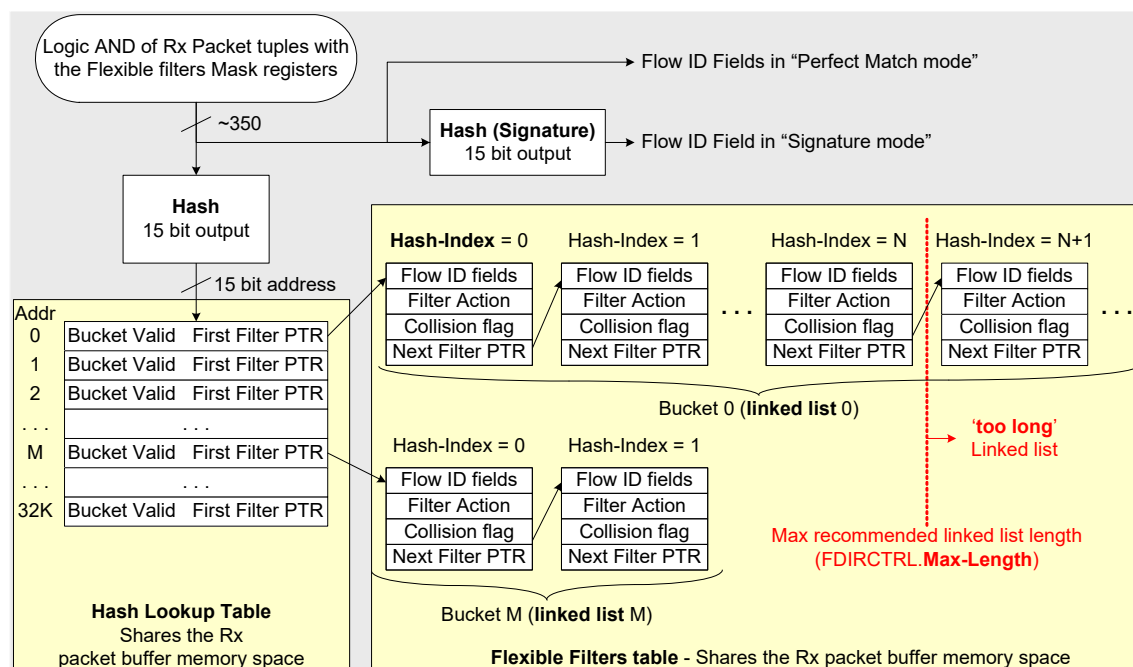


Figure 7-7. Flow Director Filters Block Diagram

7.1.3.5.5 Rx Packet Buffer Allocation

Flow director filters can consume zero space (when disabled) up to ~256 KB of memory. As shown in [Figure 7-7](#), flow director filters share the same memory with the Rx packet buffer. By setting the *PBALLOC* field in the *FDIRCTRL* register, the software can enable and allocate memory for the flow director filters. The memory allocated to received traffic is the remaining part of the Rx packet buffer.

Table 7-8. Rx Packet Buffer Allocation

PBALLOC (2)	Effective Rx Packet Buffer Size (see following note)	Flow Director Filters Memory	Supported Flow Director Filters			
			Signature		Perfect Match	
			Filters	Bucket Hash	Filters	Bucket Hash
00 Flow Director is disabled	384 KB	0	0	N/A	0	N/A
01	320 KB	64 KB	8 K -1 filters	13 bit	2 K -1 filters	11 bit
10	256 KB	128 KB	16 K -1 filters	14 bit	4 K -1 filters	12 bit
11	128 KB	256 KB	32 K -1 filters	15 bit	8 K -1 filters	13 bit

Notes:

- It is the user's responsibility to ensure that sufficient buffer space is left for reception of traffic. The required buffer space for receive traffic depends on the number of traffic classes, and flow control upper and lower threshold values. If flow director is enabled (*PBALLOC* > 0), software should set the *RXPBSIZE[n]* registers according to the total remaining part of the Rx packet buffer for reception of traffic.
- When allocating 256 KB to the Flow Director table, it is likely that the 128 KB packet buffer left is too small to permit operating the port in DCB enabled mode.
- Refer to [Section 3.6.5.3.2](#) through [Section 3.6.5.3.5](#) for recommended setting of the Rx packet buffer sizes and flow control thresholds.

7.1.3.5.6 Flow Director Filtering Reception Flow

- Rx packet is digested by the filter unit which parse the packet extracting the relevant tuples for the filtering functionality.
- The E610 calculates a 15-bit hash value out of the masked tuples (logic mask of the tuples and the relevant mask registers) using the hash function described in [Section 7.1.3.5.16](#).
- The address in the hash lookup table points to the selected linked list of flow director filters.
- The E610 checks the *Bucket Valid* flag. If it is inactive, the packet does not match any filter. Otherwise, *Bucket Valid* flag is active, proceed for the next steps.
- The E610 checks the linked list until it reaches the last filter in the linked list or until a matched filter is found.
- Case 1: Matched filter is found:
 - Increment the *FDIRMATCH* statistic counter.
 - Process the filter's actions (queue assignment) according to queue assignment priority. Meaning, the actions defined in this filter takes place only if the packet did not match any L2 filter or SYN filter that assigns an Rx queue to the packet.
 - Rx queue assignment according to the filter context takes place if *QUEUE_EN* is set.
 - If the *DROP* bit is set in the filter context, the packet is sent to the queue pointed by the *FDIRCTRL.DROP_QUEUE* field.

- Post the packet to host including the flow director filter match indications as described in [Section 7.1.3.5.3](#).
- Case 2: Matched filter is not found:
 - Increment the FDIRMISS statistic counter.
 - Post the packet to host including the flow director filter miss indications as described in [Section 7.1.3.5.3](#).

7.1.3.5.7 Add Filter Flow

The software programs the filters parameters in the registers described in [Section 7.1.3.5.13](#) and [Section 7.1.3.5.14](#) while keeping the FDIRCMD.FILTER_UPDATE bit inactive. As a result, the E610 checks the bucket valid indication in the hash lookup table (that matches the FDIRHASH.HASH) for the presence of an existing linked list. Following are the two programming flows for the case a link list exists or for the case a new list is required.

- Case 1: Add a filter to existing linked list:

The E610 checks the linked list until it reaches the last filter in the list or until a matched filter is found. The following cases may occur:

- Matched filter is found (equal flow ID) with the same action parameters — The programming is discarded silently. This is a successful case since the programmed flow is treated as requested.
 - Matched filter is found (equal flow ID) with different action parameters — The E610 keeps the old setting of the filter while setting the *Collision* flag in the filter context (see [Section 7.1.3.5.3](#) for software handling of collision during packet reception).
 - Matched filter is found (equal flow ID) with different action parameters and the *Collision* flag is already set — The programming is discarded silently. Software gets the same indications as the previous case.
 - Matched filter is not found (no collision) — The E610 checks for a free space in the flow director filters table.
 - No space case — Discard programming; increment the *FADD* counter in the FDIRFSTAT register and assert the flow director interrupt. Following this interrupt software should read the FDIRFSTAT register and FDIRFREE.FREE field, for checking the interrupt cause.
 - Free space is found — Good programming case: Add the new filter at the end of the linked list while indicating it as the last one. Program the *Next Filter PTR* field and then clear the *Last* flag in the filter that was previously the last one.
- Case 2: Create a new linked list:

The E610 looks for an empty space in the flow director filters table:

 - Handle no empty space the same as in Case 1.
 - Good programming case: Add the new filter while indicating it as the last one in the linked list. Then, program the hash lookup table entry by setting the *Valid* flag and the *First Filter PTR* pointing to the new programmed filter.

Additional successful add flow indications:

- Increment the *ADD* statistic counter in the FDIRUSTAT register.
- Reduce the *FREE* counter in the FDIRFREE register, and then indicate the number of free filters. If the *FREE* counter crosses the *FULL_THRESH* value in the FDIRCTRL register, assert the flow director filter interrupt. Following this interrupt software should read the FDIRFSTAT register and FDIRFREE.*FREE* field, for checking the interrupt cause.
- Compare the length of the new linked list with *MAXLEN* in the FDIRLEN register. If the new linked list is longer than *MAXLEN*, update the FDIRLEN by the new flow.

7.1.3.5.8 Update Filter Flow

In some applications, it is useful to update the filter parameters, such as the destination Rx queue. Programming filter parameters is described in [Section 7.1.3.5.7](#).

Setting the *FILTER_UPDATE* bit in the FDIRCMD register has the following result:

- Case 1: Matched filter does not exist in the filter table — Setting the *FILTER_UPDATE* bit has no impact and the command is treated as add filter.
- Case 2: Matched filter already exists in the filter table — Setting the *FILTER_UPDATE* bit enables filter parameter's update while keeping the collision indication as is.

When updating an existing filter the software device driver should program the same filter (i.e. the same FDIRHASH.*HASH* and the same FDIRHASH.*SIGNATURE_SW_INDEX*) while keeping the *FDIRCMD.FILTER_UPDATE* = 1

7.1.3.5.9 Remove Filter Flow

Software programs the filter Hash and Signature/Software-Index in the FDIRHASH register. It then should set the FDIRCMD.*CMD* field to *Remove Flow*. Software might use a single 64-bit access to the two registers for atomic operation. As a result, the E610 follows these steps:

1. Check if such a filter exists in the flow director filters table.
2. If there is no flow, increment the *FREMOVE* counter in the FDIRFSTAT register and skip the next steps.
3. If the requested filter is the only filter in the linked list, invalidate its entry in the hash lookup table by clearing the *Valid* bit.
4. Else, if the requested filter is the last filter in the linked list, invalidate the entry by setting the *Last* flag in the previous filter in the linked list.
5. Else, invalidate its entry by programming the *Next Filter PTR* in the previous filter in the linked list, pointing it to the filter that was linked to the removed filter.

Additional indications for successful filter removal:

1. Increment the remove statistic counter in the FDIRUSTAT register.
2. Increment the *FREE* counter in the FDIRFREE register.

7.1.3.5.10 Remove All Flow Director Filters

In some cases there is a need to clear the entire flow director table. It might be useful in some applications that might cause the flow director table becoming too occupied. Then, software might clear the entire table enabling its re-programming with new active flows.

Following are steps required to clear the flow director table:

- Poll the `FDIRCMD.CMD` until it is zero indicating any previous pending commands to the flow director table is completed (at worst case the `FDIRCMD.CMD` should be found cleared on the second read cycle). Note that software must not initiate any additional commands (add/remove/query) before this step starts and until this flow completes.
- Clear the `FDIRFREE` register (set `FREE` field to zero).
- Set `FDIRCMD.CLEARHT` to 1b and then clear it back to 0b.
- Clear the `FDIRHASH` register to zero.
- Re-write `FDIRCTRL` by its previous value while clearing the `INIT_DONE` flag.
- Poll the `INIT_DONE` flag until it is set to 1b by hardware.
- Clear the following statistic registers: `FDIRUSTAT`, `FDIRFSTAT`, `FDIRMATCH`, `DIRMISS`, `DIRLEN` (note that some of these registers are read clear and some are read write).

7.1.3.5.11 Flow Director Filters Initializing Flow

Following a device reset, the flow director is enabled by programming the `FDIRCTRL` register, as follows:

- Set `PBALLOC` to non-zero value according to the required buffer allocation to reception and flow director filter (see [Section 7.1.3.5.5](#)). All other fields in the register should be valid as well (according to required setting), as the `FDIRCTRL` register is expected to be programmed by a single cycle.
- Poll the `INIT_DONE` flag until it is set to 1b by hardware (expected initialization flow should take about 55 μ s at 10 Gb/s and 550 μ s at 1 Gb/s).

Note: The value of the `FDIRCTRL.PERFECT_MATCH` and `FDIRCTRL.FILTERMODE` should be modified only when no filters are set (at init time).

7.1.3.5.12 Query Filter Flow

Software might query specific filter settings and bucket length using the Query command.

- Program the filter `HASH` and `SIGNATURE_SW_INDEX` in the `FDIRHASH` register, and set the `CMD` field in the `FDIRCMD` register to 11b (Query Command). A single 64-bit access can be used for this step.
- As a result, the E610 provides the query result in the `FDIRHASH`, `FDIRCMD` and `DIRLEN` registers (described in the sections as follows).
- Hardware indicates query completion by clearing the `FDIRCMD.CMD` field. The following table lists the query result.

Table 7-9. Query Filter Flow

Query Outcome	FDIRHASH -> Bucket Valid	FDIRCMD -> Filter Valid	FDIRLEN -> Bucket Length	FDIRCMD -> Filter ID Fields	FDIRCMD -> Filter Action
Empty Bucket	0	0	N/A	N/A	N/A
Valid Bucket, Matched Filter Not Found	1	0	Bucket linked list length	N/A	N/A
Found Signature Filter	1	1	Filter index within the linked list	0	Filter's parameters ¹
Found Perfect Match Filter	1	1	Filter index within the linked list	Filter's parameters	Filter's parameters

1. The pool parameter is not returned as part of the filter action in signature mode.

7.1.3.5.13 Signature Filter Registers

The signature flow director filter is programmed by setting the FDIRHASH and FDIRCMD registers. These registers are located in consecutive 8-byte aligned addresses. Software should use a 64-bit register to set these two registers in a single atomic operation. [Table 7-10](#) lists the recommended setting.

Table 7-10. Signature Match Filter Parameters

Filter Bucket Parameters — FDIRHASH	
Hash	15-bit hash function used to define a bucket of filters. This parameter is part of the flow director filter ID that can be reported in the Rx descriptor. It is shared for signature and perfect match filters.
Valid	Should be set to 1b. It is shared for signature and perfect match filters.
Flow ID — FDIRHASH	
Signature	15-bit hash function used as the flow matching field. This parameter is also part of the flow director filter ID that can be reported in the Rx descriptor.
FDIRCMD — Programming Command and Filter action — See Section 8.2.2.14.23 for all fields descriptions.	

7.1.3.5.14 Perfect Match Filter Registers

Perfect match filters are programmed by the following registers: FDIRSIPv6[n], FDIRVLAN, FDIRPORT, FDIRIPDA, FDIRIPSA, FDIRHASH, FDIRCMD. Setting the FDIRCMD register, generates the actual programming of the filter. Therefore, write access to this register must be the last cycle after all other registers contain a valid content. [Table 7-11](#) lists the recommended setting.

Note: Software filter programming must be an atomic operation. In a multi-core environment, software must ensure that all registers are programmed in a sequence with no possible interference by other cores.

Table 7-11. Perfect Match Filter Parameters

Filter Bucket Parameters and Software Index — FDIRHASH			
Hash	See Section 7.1.3.5.13 .		
Valid	See Section 7.1.3.5.13 .		
Software-Index	16-bit index provided by software at filter programming used by software to identify the matched flow. This parameter is also part of the flow director filter ID that can be reported in the Rx descriptor.		
FDIRCMD — Programming Command and Filter Action Set			
	IP Mode <i>FILTERMODE = 000b</i>	MAC, VLAN Mode <i>FILTERMODE = 001b</i>	Cloud Modes <i>FILTERMODE = 010b</i>
IPV6DMATCH	Should be set for IPv6 filters only and only if the destination address should be compared to one of the IPAT filters.	Should be cleared	
L4TYPE	Defines if the filter is for TCP, UDP or SCTP flows.	Should be cleared FDIRM.L4P should be set	
IPV6	Defines if the filter is for IPv4 or IPv6 addresses.	Should be cleared FDIRM.L3P should be set FDIRM.DIPV6 should be set	
TUNNEL_FILTER	Defines if the filter is a VXLAN and NVGRE tunneled packet filter. If set, the parameters of the L3 and L4 header relates to the tunneled (inner) header.	Should be cleared.	
POOL	Pool to which the packet is directed (relevant only in virtualization modes).		
Flow ID — Perfect Match Flow ID Parameters are Listed in the Following Registers and Fields			
	IP Mode <i>FILTERMODE = 000b</i>	MAC, VLAN Mode <i>FILTERMODE = 001b</i>	Cloud Mode <i>FILTERMODE = 010b</i>
FDIRSIPv6[0...2].IP6SA	Three LS DWord of the source IPv6. Meaningful for IPv6 flows depending on the FDIRIP6M.SIPM setting.	{FDIRSIPv6_0[31:0], FDIRSIPv6_1[15:0]} holds the destination MAC Address value FDIRSIPv6_1[31:16], and FDIRSIPv6_2[31:0] are reserved and should be set to 0x0. The FDIRIP6M.SIPM field should be set to 0x0xFC0F	FDIRSIPv6_2[31:0] holds the 32 bits of the <i>TNI/VNI</i> field. {FDIRSIPv6_0[31:0], FDIRSIPv6_1[15:0]} holds the destination MAC Address value FDIRSIPv6_1[31] holds the cloud mode (0 = NVGRE, 1 = VXLAN). FDIRSIPv6_1[30:16] are reserved and should be set to 0x0. To mask one of the fields the matching bits in FDIRIP6M.SIPM should be set: • Bits [15:12] control the masking of the <i>TNI/VNI</i> field. All the 32 bits of the key can be used for filtering. • Bit [11] controls the masking of the cloud mode and should be cleared for normal operation • Bits [3:0] and Bit [10] should be set. • Bits [9:4] control the masking of the inner MAC Address.

Table 7-11. Perfect Match Filter Parameters [continued]

FDIRVLAN.VLAN	VLAN fields are meaningful depending on the FDIRM.VLANID and FDIRM.VLANP setting. The VLAN field should be stored in Little endian format.	Inner VLAN fields are meaningful depending on the FDIRM.VLANID and FDIRM.VLANP setting. The VLAN field should be stored in Little endian format.
FDIRVLAN.FLEX	Flexible 2-byte field at offset FDIRCTRL.FLEX_OFFSET. Meaningful depending on FDIRM.FLEX setting.	
FDIRPORT.Source	L4 source port. Meaningful for TCP, UDP and SCTP packets depending on the FDIRTCPM.SportM, FDIRUDPM.SportM and FDIRSCTPM.SportM setting.	Reserved - Should be zero. FDIRM.L4P should be set. FDIRTCPM.SportM, FDIRUDPM.SportM and FDIRSCTPM.SportM should be set.
FDIRPORT.Destination	L4 destination port. Meaningful for TCP, UDP and SCTP packets depending on the FDIRTCPM.DportM, FDIRUDPM.DportM and FDIRSCTPM.DPORTM setting.	Reserved - Should be zero. FDIRM.L4P should be set. FDIRTCPM.DportM, FDIRUDPM.DportM and FDIRSCTPM.DportM should be set.
FDIRIPDA.IP4DA	IPv4 destination address. Meaningful depending on the FDIRIP4M.IPM setting.	Reserved - Should be zero. All bits in FDIRIP4M.IPM should be set.
FDIRIPSA.IP4SA	IPv4 source address or LS DWord of the source IPv6 address. Meaningful for IPv4 flows depending on the FDIRSIP4M.IPM setting and for IPv6 flows depending on the FDIRIP6M.SIPM setting.	Reserved - Should be zero. All bits in FDIRSIP4M.IPM should be set. FDIRIP6M.SIPM[3:0] should be set.

7.1.3.5.15 Multiple CPU Cores Considerations

Perfect match filters programming and any query cycles require access to multiple registers. To avoid races between multiple cores, software might need to use one of the following programming methods:

- Use a software-based semaphore between the multiple cores for gaining control over the relevant CSR registers for complete programming or query cycles.
- Manage all programming and queries of the flow director filters by a single core.

Programming signature filters requires only the FDIRHASH and FDIRCMD registers. These two registers are located in 8-byte aligned adjacent addresses. Software could use an 8-byte register for the programming of these registers in a single atomic operation, which avoids the need for any semaphore between multiple cores.

7.1.3.5.16 Flow Director Hash Function

The E610 supports programmable 16-bit hash functions based on two 32-bit keys, one for the lookup table identifying a bucket of filters and another one for the signature (*FDIRHKEY* and *FDIRSKEY*). The hash function is described in the sections that follows. In some cases, a smaller hash value than 16 bits is required. In such cases, the LS bits of the hash value are used.

```
For (i=0 to 350) {if (Ext_K[i]) then Hash[15: 0] = Hash[15: 0] XOR Ext_S[15+i: i]}
```


While using the following notations:

'XOR'	Bitwise XOR of two equal length strings
If (xxx)	Equals 'true' if xxx = '1' and equals 'false' if xxx = '0'
S[335:0]	The input bit string of the flow director tuples: 42 bytes listed in Table 7-12 AND-logic with the filters masks.
Ext_S[n]	S[14:0] S[335:0] S[335:321] // concatenated
K[31:0]	The hash key as defined by the FDIRHKEY or FDIRSKEY registers.
Tmp_K[11*32-1:0]	(Temp Key) equals K[31:0] K[31:0]... // concatenated Key 11 times
Ext_K[350:0]	(Extended Key) equals T_K[351:1]

The input bit stream for the hash calculation is listed in [Table 7-12](#) while byte 0 is the MSByte (first on the wire) of the VLAN, byte 2 is the MSByte of the source IP (IPv6 case) and so on.

Table 7-12. Input Bit Stream for Hash Calculation

Bytes/Mode	Field		
	IP	MAC, VLAN	Cloud Mode
Bytes 0...1	VLAN tag (always the outer VLAN)		Inner VLAN Tag
Bytes 2...5	Source IP (16 bytes for IPv6; source IP for IPv4 12 bytes of zero's)	Zero	
Bytes 6...11		MAC	Inner MAC
Bytes 12...13		Zero	8 bits of zeros bit of tunnel_type 7 bits of zeros
Bytes 14...17		Zero	VNI/TNI
Bytes 18...33	Signature Mode: Destination IP (16 bytes for IPv6; destination IP for IPv4 12 bytes of zero's) Exact Mode: Destination IP (7 zero bits IP6AT match indication 120 zero bits; destination IP for IPv4 12 bytes of zero's) ¹		
34...37	L4 source port number L4 destination port number		
38...39	Flexible bytes		
40	00b pool number (as defined by FDIRCMD.Pool)		
41	[7:5] 000b [4] FDIRCMD.TUNNEL_FILTER (should be zero in MAC, VLAN mode). [3] 0b [2] IPv6/IPv4 type (FDIRCMD.IPV6) [1:0] L4 type (FDIRCMD.L4TYPE)		

1. In VXLAN and NVGRE packets, the IP Addresses used are of the tunneled header in other packets the outer IP header is used.

7.1.3.6 RSS

RSS is a mechanism to distribute received packets into several descriptor queues. Software can then assign each queue to a different processor, therefore sharing the load of packet processing among several processors.

As described in [Section 7.1](#), the E610 uses RSS as one ingredient in its packet assignment policy (the others are the various filters, DCB and virtualization). The RSS output is an RSS index. The E610 global assignment uses these bits (or only some of the LSBs) as part of the queue number.

Figure 7-8 and Figure 7-9 show the process of computing an RSS output:

1. The receive packet is parsed into the header fields used by the hash operation (such as IP Addresses, TCP port, etc.)
2. A hash calculation is performed. The E610 supports a single hash function, as defined by Microsoft* (MSFT) RSS. The E610 therefore does not indicate to the device driver which hash function is used. The 32-bit result is fed into the *RSS Hash* field in the packet receive descriptor.
3. The E610 supports two modes of RSS defined by the MRQC.*MULTIPLE_RSS* bit. When set to 0b a single RSS key and redirection table are supported. In this mode (usually used when virtualization is not enabled) the nine LSBs of the hash result are used as an index into a 512-entry redirection table. Each entry provides up to 6-bit RSS output index (Figure 7-8).

For SRIOV or VMDQ enabled modes (set by MRQC.*MULTIPLE_RSS* set to 1b), The E610 supports up to 64 (one per pool) RSS keys and redirection tables (both can be controlled and programmed at the VF space). In this mode the 6 LSBs of the hash result are used as an index into a 64-entry redirection table. Each entry provides up to 2-bit RSS output index (Figure 7-9).

When RSS is enabled, the E610 provides software with the following information as required by Microsoft* RSS and provided for device driver assist:

- A DWord result of the MSFT RSS hash function application to the packet header, to be used by the stack for flow classification, is written into the receive packet descriptor. A 4-bit *RSS Type* field conveys the hash function used for the specific packet.
- Packets to which the RSS function cannot be applied (for example non IP packets) return an *RSS Hash* result of zero, an *RSS Type* of zero and an RSS output index of zero.

7.1.3.6.1 Enabling RSS

- RSS is enabled in the MRQC register.
- RSS enabling cannot be done dynamically and must be preceded by a software reset.
- RSS status field in the descriptor write-back is enabled when the RXCSUM.*PCSD* bit is set (fragment checksum is disabled). RSS is therefore mutually exclusive with UDP fragmentation checksum offload.
- Support for RSS is not provided when legacy receive descriptor format is used.

7.1.3.6.2 Disabling RSS

- Disabling RSS on the fly is not allowed, and the E610 must be reset after RSS is disabled.
- When RSS is disabled, packets are assigned an RSS output index = zero.

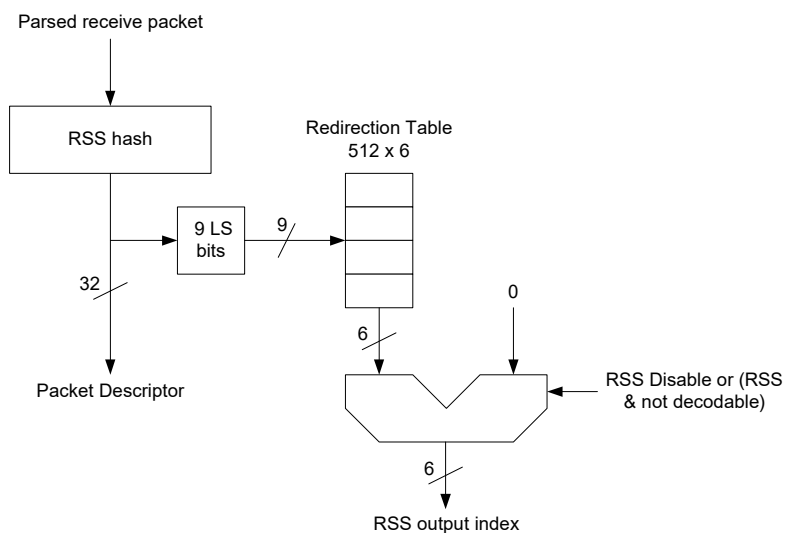


Figure 7-8. RSS Block Diagram (***MULTIPLE_RSS*** = 0b)

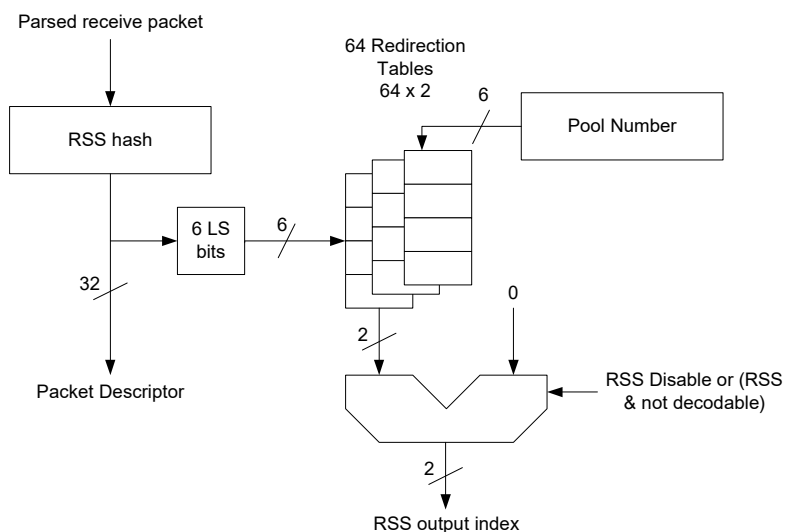


Figure 7-9. RSS Block Diagram (***MULTIPLE_RSS*** = 1b)

7.1.3.6.3 RSS Hash Function

This section provides a verification suite used to validate that the hash function is computed according to MSFT nomenclature.

The E610's hash function follows the MSFT definition. A single hash function is defined with several variations for the following cases:

- **TcpIPv4** — The E610 parses the packet to identify an IPv4 packet containing a TCP segment per the criteria described below. If the packet is not an IPv4 packet containing a TCP segment, RSS is not done for the packet.

- **IPv4** — The E610 parses the packet to identify an IPv4 packet. If the packet is not an IPv4 packet, RSS is not done for the packet.
- **TcpIPv6** — The E610 parses the packet to identify an IPv6 packet containing a TCP segment per the criteria described below. If the packet is not an IPv6 packet containing a TCP segment, RSS is not done for the packet.
- **IPv6** — The E610 parses the packet to identify an IPv6 packet. If the packet is not an IPv6 packet, RSS is not done for the packet.

Tunneled IP to IP packets are considered for the RSS functionality as IP packets. The RSS logic ignores the L4 header while using the outer (first) IP header for the RSS hash.

For NVGRE and VXLAN packets the inner header (IP and TCP/UDP) is used for RSS computations.

The following additional cases are not part of the MSFT RSS specification but are supported RSS modes:

- **UdpIPv4** — The E610 parses the packet to identify a packet with UDP over IPv4.
- **UdpIPv6** — The E610 parses the packet to identify a packet with UDP over IPv6.

A packet is identified as containing a TCP segment if all of the following conditions are met:

- The transport layer protocol is TCP (not UDP, ICMP, IGMP, etc.).
- The TCP segment can be parsed (such as IPv4 options or IPv6 extensions can be parsed, packet not encrypted, etc.).
- The packet is not fragmented (even if the fragment contains a complete L4 header).

Note: IPv6 extended headers are parsed by the E610, enabling TCP layer header recognition. Still the IPv6 extended header fields are not taken into account for the queue classification by RSS filter. This rule do not apply for security headers and fragmentation header. Packets with fragmentation header miss this filter. Packets with security extended headers are parsed only up to these headers and therefore can match only filters that do not require fields from the L4 protocol.

Bits[31:16] of the Multiple Receive Queues Command (MRQC), for single RSS and VFMRQC for multiple RSS, registers enable each of the above hash function variations (several might be set at a given time). If several functions are enabled at the same time, priority is defined as follows (skip functions that are not enabled):

- IPv4 packet:
 - Try using the TcpIPv4 function
 - Try using UdpIPv4 function
 - Try using the IPv4 function
- IPv6 packet:
 - Try using the TcpIPv6 function.
 - Try using UdpIPv6 function.
 - Try using the IPv6 function

The following combinations are currently supported:

- Any combination of IPv4, TcpIPv4, and UdpIPv4.
And/or:
- Any combination of either IPv6, TcpIPv6, and UdpIPv6.

When a packet cannot be parsed by the previous rules, it is assigned an RSS output index = zero. The 32-bit tag (normally a result of the hash function) equals zero.

The 32-bit result of the hash computation is written into the packet descriptor and also provides an index into the redirection table.

The following notation is used to describe the following hash functions:

- Ordering is little endian in both bytes and bits. For example, the IP Address 161.142.100.80 translates into 0xa18e6450 in the signature.
- A “^” denotes bit-wise XOR operation of same-width vectors.
- @x-y denotes bytes x through y (including both of them) of the incoming packet, where byte 0 is the first byte of the IP header. In other words, we consider all byte-offsets as offsets into a packet where the framing layer header has been stripped out. Therefore, the source IPv4 address is referred to as @12-15, while the destination v4 address is referred to as @16-19.
- @x-y, @v-w denotes concatenation of bytes x-y, followed by bytes v-w, preserving the order in which they occurred in the packet.

All hash function variations (IPv4 and IPv6) follow the same general structure. Specific details for each variation are described in the following section. The hash uses a random secret key of length 320 bits (40 bytes); the key is stored in the RSS Random Key Register RSSRK for single RSS and VFRSSRK[63:0] for multiple RSS.

The algorithm works by examining each bit of the hash input from left to right. Our nomenclature defines left and right for a byte-array as follows: Given an array K with k bytes, our nomenclature assumes that the array is laid out as follows:

- K[0] K[1] K[2] ... K[k-1]

K[0] is the left-most byte, and the MSB of K[0] is the left-most bit. K[k-1] is the right-most byte, and the LSB of K[k-1] is the right-most bit.

ComputeHash(input[], N)

```
For hash-input input[] of length N bytes (8N bits) and a random secret key K of 320 bits
Result = 0;
For each bit b in input[] {
  if (b == 1) then Result ^= (left-most 32 bits of K);
  shift K left 1 bit position;
}
return Result;
```

7.1.3.6.3.1 Pseudo-code Examples

The following four pseudo-code examples are intended to help clarify exactly how the hash is to be performed in four cases, IPv4 with and without ability to parse the TCP header, and IPv6 with and without a TCP header.

Hash for IPv4 with TCP:

Concatenate SourceAddress, DestinationAddress, SourcePort, DestinationPort into one single byte-array, preserving the order in which they occurred in the packet: Input[12] = @12-15, @16-19, @20-21, @22-23.

```
Result = ComputeHash(Input, 12);
```

Hash for IPv4 with UDP:

Concatenate SourceAddress, DestinationAddress, SourcePort, DestinationPort into one single byte-array, preserving the order in which they occurred in the packet: Input[12] = @12-15, @16-19, @20-21, @22-23.

```
Result = ComputeHash(Input, 12);
```

Hash for IPv4 without TCP:

Concatenate SourceAddress and DestinationAddress into one single byte-array

```
Input[8] = @12-15, @16-19
Result = ComputeHash(Input, 8)
```

Hash for IPv6 with TCP:

Similar to previous:

```
Input[36] = @8-23, @24-39, @40-41, @42-43
Result = ComputeHash(Input, 36)
```

Hash for IPv6 with UDP:

Similar to previous:

```
Input[36] = @8-23, @24-39, @40-41, @42-43
Result = ComputeHash(Input, 36)
```

Hash for IPv6 without TCPL:

```
Input[32] = @8-23, @24-39
Result = ComputeHash(Input, 32)
```

7.1.3.6.4 Redirection Tables

The redirection table used for single RSS mode (RETA - [Section 8.2.2.8.18](#) and ERETA - [Section 8.2.2.8.19](#)) is a 512-entry structure, indexed by the nine LSBs of the hash function output.

The redirection tables used for multiple RSS mode (VFRETA - [Section 8.3.3.3.11](#)) are 64-entry structures indexed by the six LSBs of the hash function output. The table to use is defined by the pool to which the packet is sent.

System software might update the redirection tables during run time. Such updates of the table are not synchronized with the arrival time of received packets. Therefore, it is not guaranteed that a table update takes effect on a specific packet boundary.

7.1.3.6.5 RSS Verification Suite

Assume that the random key byte-stream is:

```
0x6d, 0x5a, 0x56, 0xda, 0x25, 0x5b, 0x0e, 0xc2,
0x41, 0x67, 0x25, 0x3d, 0x43, 0xa3, 0x8f, 0xb0,
0xd0, 0xca, 0x2b, 0xcb, 0xae, 0x7b, 0x30, 0xb4,
0x77, 0xcb, 0x2d, 0xa3, 0x80, 0x30, 0xf2, 0x0c,
0x6a, 0x42, 0xb7, 0x3b, 0xbe, 0xac, 0x01, 0xfa
```

Table 7-13. IPv4

Destination Address/Port	Source Address/Port	IPv4 only	IPv4 with TCP
161.142.100.80:1766	66.9.149.187:2794	0x323e8fc2	0x51ccc178
65.69.140.83:4739	199.92.111.2:14230	0xd718262a	0xc626b0ea
12.22.207.184:38024	24.19.198.95:12898	0xd2d0a5de	0x5c2b394a
209.142.163.6:2217	38.27.205.30:48228	0x82989176	0xafc7327f
202.188.127.2:1303	153.39.163.191:44251	0x5d1809c5	0x10e828a2

Note: The IPv6 address tuples are only for verification purposes, and may not make sense as a tuple.

Table 7-14. IPv6

Destination Address/Port	Source Address/Port	IPv6 only	IPv6 with TCP
3ffe:2501:200:3::1 (1766)	3ffe:2501:200:1fff::7 (2794)	0x2cc18cd5	0x40207d3d
ff02::1 (4739)	3ffe:501:8::260:97ff:fe40:efab (14230)	0x0f0c461c	0xdde51bbf
fe80::200:f8ff:fe21:67cf (38024)	3ffe:1900:4545:3:200:f8ff:fe21:67cf (44251)	0x4b61e985	0x02d1feef

7.1.4 Receive Data Storage in System Memory

The E610 posts receive packets into data buffers in system memory.

The following controls are provided for the data buffers:

- The `SRRCTL[n].BSIZEPACKET` field defines the size of the data buffer pointed by each descriptor. The maximum packet size that can be posted to a queue can be limited using the `RXDCTL[n].RLPML` field. This filter enables software to use smaller buffers than the size defined by the `SRRCTL[n].BSIZEPACKET`.

Note: The packet size compared to `RXDCTL[n].RLPML` does not include any parts stripped by the device like CRC VLAN or other tags but include additional Timestamp appended to the packet.

- The `SRRCTL[n].BSIZEHEADER` field defines the size of the header buffer pointed by each descriptor (advanced descriptors only).
- Each queue is provided with a separate `SRRCTL` register.

Receive memory buffer addresses are word (2 x byte) aligned (both data and headers).

The internal receive buffers are described in [Section 7.6.3.1](#).

7.1.5 Receive Descriptors

7.1.5.1 Legacy Receive Descriptor Format

A receive descriptor is a data structure that contains the receive data buffer address and fields for hardware to store packet information. Upon receipt of a packet for this device, hardware stores the packet data into the indicated buffer and writes the length, status and errors to the receive descriptor. If `SRCTL[n].DESCTYPE = zero`, the E610 uses the Legacy Rx descriptor as listed in [Table 7-15](#). The shaded areas indicate fields that are modified by hardware upon packet reception (so-called descriptor write-back).

Legacy descriptors should not be used when advanced features are enabled: SCTP, Virtualization, DCB, Time stamp in Packet, tunnel packets checksum. Packets that match these cases might be dropped from queues that use legacy receive descriptors.

Refer to [Table 7-15](#) and the field descriptions that follow.

Table 7-15. Legacy Receive Descriptor (RDESC) Layout

	63	48	47	40	39	32	31	16	15	0
0	Buffer Address [63:0]									
8	VLAN Tag		Errors		Status		Fragment Checksum		Length	

Buffer Address (64-bit offset 0, 1st line)

Physical address in host memory of the received packet buffer.

Length Field (16-bit offset 0, 2nd line)

The length indicated in this field covers the data written to a receive buffer including CRC bytes (if any). Software must read multiple descriptors to determine the complete length for packets that span multiple receive buffers.

Fragment Checksum (16-bit offset 16, 2nd line)

This field is used to provide the fragment checksum value. This field is equal to the unadjusted 16-bit ones complement of the packet. Checksum calculation starts at the L4 layer (after the IP header) until the end of the packet excluding the CRC bytes. To use the fragment checksum assist to offload L4 checksum verification, software might need to back out some of the bytes in the packet. For more details see [Section 7.1.6.5](#).

The fragment checksum is always reported in the descriptor with the EOP bit set.

Status Field (8-bit offset 32, 2nd line)

Status information indicates whether the descriptor has been used and whether the referenced buffer is the last one for the packet. Error status information is listed in [Table 7-17](#).

Table 7-16. Receive Status (RDESC.STATUS) Layout

7	6	5	4	3	2	1	0
PIF	IPCS	L4CS	UDPCS	VP	Reserved	EOP	DD

End of Packet (EOP) and Descriptor Done (DD)

Refer to the following table:

DD	EOP	Description
0	0	Software setting of the descriptor when it hands it to the hardware.
0	1	Reserved (invalid option).
1	0	A completion status indication for non-last descriptor of a packet that spans across multiple descriptors. It means that the hardware is done with the descriptor and its buffers while only the length is valid on this descriptor.
1	1	A completion status indication of the entire packet. Software might take ownership of its descriptors while all fields in the descriptor are valid.

VP (VLAN Packet)

When set, the *VP* field indicates that the incoming packet's type is a VLAN (802.1q, matching the *VLNCTRL.VET*). If the *RXDCTL.VME* bit is set as well, an active *VP* field also means that the VLAN has been stripped from the packet to the receive descriptor. For a further description of 802.1q VLANs, see [Section 7.4](#).

IPCS (IPv4 Checksum), L4CS (L4 Checksum), UDPCS (UDP Checksum)

These bits are described in the following table.

Note: Switched packets from a local VM that do not use the Tx IP checksum offload by hardware have the *IPCS* equal to zero; switched packets from a local VM that do not use the Tx L4 checksum offload by hardware have the *L4CS* and *UDPCS* equal to zero.

L4CS	UDPCS	IPCS	Functionality
0	0	0	Hardware does not provide checksum offload.
0	0	1	Hardware provides IPv4 checksum offload. Pass/fail indication is provided in the <i>Error</i> field – <i>IPE</i> .
1	0	1/0	Hardware provides IPv4 checksum offload if <i>IPCS</i> is active along with TCP checksum offload. Pass/fail indication is provided in the <i>Error</i> field – <i>IPE</i> and <i>TCPE</i> .
1	1	1/0	Hardware provides IPv4 checksum offload if <i>IPCS</i> is active along with UDP checksum offload. Pass/fail indication is provided in the <i>Error</i> field – <i>IPE</i> and <i>TCPE</i> .

IPv6 packets do not have the *IPCS* bit set, but might have the *L4CS* bit and *UDPCS* bit set if the E610 recognizes the transport header.

PIF (Non Unicast Address)

The *PIF* bit is set on packets with a non-unicast destination Ethernet MAC Address — multicast and broadcast.

Error Field (8-bit offset 40, 2nd line)

[Table 7-17](#) and the following text describes the possible errors reported by the hardware.

Table 7-17. Receive Errors (RDESC.ERRORS) Layout

7	6	5	4	3	2	1	0
IPE	TCPE	Reserved	Reserved	Reserved	Reserved	Reserved	RXE

IPE (IPv4 Checksum Error)

The IP checksum error is valid only when the *IPCS* bit in the *Status* field is set (indicating that the hardware validated the IP checksum). This bit is meaningful only on the last descriptor of a packet while the *EOP* bit is set as well. Packets with IP error are posted to host memory regardless of the store bad packet setting (*FCTRL.SBP*).

TCPE (TCP/UDP Checksum Error)

The TCP/UDP checksum error is valid only when the *L4CS* bit in the *Status* field is set (indicating that the hardware validated the L4 checksum). This bit is meaningful only on the last descriptor of a packet while the *EOP* bit is set as well. Packets with a TCP/UDP error are posted to host memory regardless of the store bad packet setting (*FCTRL.SBP*).

IPv4/UDP packets that carry a null UDP checksum field are reported with *L4CS* = 0 and *TCPE* = 0 (valid packet with no checksum).

IPv6/UDP packets that carry a null UDP checksum field are reported with *L4CS* = 1 and *TCPE* = 1 (UDP checksum is mandatory over IPv6).

RXE

The RXE error bit is an indication for any MAC error. It is a logic OR function of the following errors:

- CRC or symbol error might be a result of receiving a /V/ symbol on the TBI interface, /FE/ symbol on the GMII/XGMII interface, RX_ER assertion on GMII interface, bad EOP or loss of sync during packet reception.
- Undersize frames shorter than 64 bytes.
- Oversize frames larger than the *MFS* definition in the *MAXFRS* register.
- Length error in 802.3 packet format. *Length* field is not checked in presence of a **VLAN or E-Tag**.

Packets with an RXE error are posted to host memory only when store bad packet bit (*FCTRL.SBP*) is set.

VLAN Tag Field (16-bit offset 48, 2nd line)

If the *RXDCTL.VME* is set and the received packet type is 802.1q (as defined by *VLNCTRL.VET*), the VLAN header is stripped from the packet data storage. In this case the 16 bits of the VLAN tag, priority tag and DEI from the received packet are posted to the *VLAN Tag* field in the receive descriptor. Otherwise, the *VLAN Tag* field contains 0x0000.

Table 7-18. VLAN Tag Field Layout (for 802.1q Packet)

15	13	12	11	0
PRI	DEI	VLAN		

Priority and DEI are part of 802.1Q specifications. The VLAN field is provided in network order.

7.1.5.2 Advanced Receive Descriptors

7.1.5.2.1 Advanced Receive Descriptors — Read Format

Table 7-19 lists the advanced receive descriptor programming by the software. The `SRRCTL[n].DESCTYPE` should be set to a value other than 000b when using the advanced descriptor format.

Table 7-19. Descriptor Read Format

	63	1	0
0	Packet Buffer Address [63:1]		
8	Header Buffer Address [63:1]		DD

Packet Buffer Address (64)

The physical address in host memory of the packet buffer.

Header Buffer Address (64)

The physical address in host memory of the header buffer with the lowest bit being Descriptor Done (DD). When a packet spans in multiple descriptors, only the header buffer of the first descriptor is used. In subsequent descriptors, only the data buffer is used.

During the programming phase, software must set the *DD* bit to zero (see the description of the *DD* bit in this section). This means that header buffer addresses are always word aligned.

Note: The E610 does not support null descriptors meaning packet or header addresses are zero.

7.1.5.2.2 Advanced Receive Descriptors — Write-Back Format

When the E610 writes back the descriptors, it uses the format listed in Table 7-20. The advanced descriptor writeback format is used when `SRRCTL[n].DESCTYPE` is set to a value other than 000b.

Table 7-20. Descriptor Write-Back Format

	63	48	47	32	31	30	21	20	17	16	4	3	0	
0	RSS Hash / Fragment Checksum / Flow Director Filters ID				SPH	HDR_LEN		RSV		Packet Type		RSS Type		
8	VLAN Tag		PKT_LEN		Extended Error				Extended Status					
	63	48	47	32	31	20		19						0

RSS Type (4-bit offset 0, 1st line)

The E610 must identify the packet type and then choose the appropriate RSS hash function to be used on the packet. The RSS type reports the packet type that was used for the RSS hash function.

RSS Type	Description
0x0	No hash computation done for this packet
0x1	HASH_TCP_IPv4
0x2	HASH_IPv4
0x3	HASH_TCP_IPv6

RSS Type	Description
0x4	Reserved
0x5	HASH_IPv6
0x6	Reserved
0x7	HASH_UDP_IPv4
0x8	HASH_UDP_IPv6
0x9 – 0xE	Reserved
0xF	Packet reports flow director filters status (set if flow director match, even if packet is forwarded by another filter).

Packet Type (13-bit at offset 4, 1st line)

The *Packet Type* field reports the packet type identified by the hardware as follows. Note that some of the fields in the receive descriptor are valid for specific packet types.

Bit Index	Bit 11 = 0	Bit 11 = 1 (L2 packet matching one of the ETQF filters)
0	IPV4 — IPv4 header present	EtherType — <i>ETQF</i> register index that matches the packet. Special types are defined for 802.1X, 1588.
1	IPV4O — IPv4 with options	
2	IPV6 — IPv6 header present	
3	IPV6E- IPv6 with extensions	Reserved.
4	TCP — TCP header present	
5	UDP — UDP header present	
6	SCTP — SCTP header	Reserved.
7	If bit 12 = 0 - Reserved If bit 12 = 1: 0b = NVGRE 1b = XLAN	
8		
9		
10	Reserved	Reserved
11	0b = non L2 packet	1b = L2 packet
12	NVGRE or VXLAN Tunnel Packet. If this bit is set, bit 7 indicates the tunnel type.	Reserved.

Note: UDP, TCP and IPv6 indications are not set in any IPv4 fragmented packet.

In virtualization mode, packets might be received from other local VMs. The E610 does not check the L5 header for these packets and does not report NFS header. If such packets carry IP tunneling (IPv4 — IPv6), they are reported as IPV4E. The packets received from local VM are indicated by the *LB* bit in the status field. To be identified, the *CC* bit in the transmit descriptor must be set and if it is a tunnel packet, the *TUNNEL.OUTERIPCS* must be set also. If bit 12 (Tunnel Packet) is set, all the L3/L4 indications (IPV4, IPV4O, IPV6, IPV6E, TCP, UDP, SCTP) reflects the inner header status.

HDR_LEN (10-bit offset 21, 1st line)

The *HDR_LEN* reflects the size of the packet header in byte units (if the header is decoded by the hardware). This field is meaningful only in the first descriptor of a packet and should be ignored in any subsequent descriptors. Header split is explained in [Section 7.1.6](#) while the packet types for this functionality are enabled by the PSRTYPE[n] registers.

Split Header — SPH (1-bit offset 31, 1st line)

When set to 1b, indicates that the hardware has found the length of the header. If set to 0b, the header buffer may be used only in split always mode. If the received header size is greater or equal to 1024 bytes, the *SPH* bit is not set and header split functionality is not supported. The *SPH* bit is meaningful in all the descriptors of a packet. See additional details on *SPH*, *PKT_LEN* and *HDR_LEN* as a function of split modes in [Table 7-25](#).

Packet Type	Header Length (Includes All Fields Up to the Field Specified)	Header Split
Un-recognized EtherType only with/without SNAP and with/without VLAN, or packets that match the L2 filters (MTQF).	VLAN header(s) if present. Else, <i>EtherType</i> field	No
Pv4 only or fragmented IPv4 with any payload including IPv4-IPv6 tunneling.	IPv4 header	Enabled
Non-fragmented IPv4, TCP / UDP / SCTP.	L4 header	Enabled
IPv4-IPv6, only or fragmented IPv4-IPv6 at IPv6 header with any payload.	IPv6 header (up to the fragment extension header if exist)	Enabled
IPv4-IPv6,TCP / UDP / SCTP.	L4 header	Enabled
VXLAN or NVGRE.	Cloud Header/L2 Header	Enabled ¹

1. If cloud split modes used (PSRTYPE15 or PSRTYPE16 are set). If not, according to internal header.

RSS Hash or Flow Director Filters ID (32-bit offset 32, 1st line) / Fragment Checksum (16-bit offset 48, 1st line)

This field has multiplexed functionality according to the received packet type (reported on the *Packet Type* field in this descriptor) and device setting.

Fragment Checksum

For fragmented UDP/IP packets, this field holds the UDP fragment checksum (described in [Section 7.1.6.5](#)) if both the RXCSUM.PCSD bit is cleared and RXCSUM.IPPCSE bit is set. This field is meaningful only for UDP packets where the *UDPV* bit in the *Extended Status* word is set. When Fragment Checksum is reported, bits 47:32 are invalid.

The checksum does not include any padding or timestamp added by the device.

RSS Hash / Flow Director Filters ID

the RXCSUM.PCSD bit is set, this field holds the RSS hash value or flow director filters ID.

- If the FDIRCTRL.REPORT_STATUS bit is set, the flow director filters ID is reported;
- If the RSS Type is non zero, the RSS hash is reported.
- Otherwise, the value is not valid.

Table 7-21. Checksum Enable/Disable

RXCSUM.PCSD	0 (Checksum Enable)	1 (Checksum Disable)
	Fragment Checksum and IP Identification are reported in the Rx Descriptor.	RSS Hash value is reported in the Rx Descriptor.

RSS Hash

The RSS hash value is required for RSS functionality as described in [Section 7.1.3.6](#). Note that the RSS hash is meaningful only for 'RSS Type' in the range 0x1 to 0x8.

Flow Director Filters ID

The flow director filters ID is reported only when the received packet matches a flow directory filter (see [Section 7.1.3.5](#)). The flow director filter ID field has a different structure for signature-based filters and perfect match filters as follows:

Filter Type	31	30	29	28	16	15	13	12	0
Hash-based Flow Director Filter ID	Rsv	Bucket Hash				Signature			
Perfect Match Flow Director Filter ID	Rsv			Hash		Rsv		SW-Index	

Bucket Hash — A hash value that identifies a flow director bucket. When the Flow Director table is smaller than 32 K filters the bucket hash is smaller than 15 bits. In this case the upper bit(s) are set to zero.

Signature — A hash value used to identify flow within a bucket.

SW-Index — The SW-Index that is taken from the filter context, programmed by software. It is meaningful only when the *FLM* bit in the Extended Status is set as well.

Rsv — Reserved.

Extended Status (20-bit offset 0, 2nd line)

Status information indicates whether the descriptor has been used and whether the referenced buffer is the last one for the packet. [Table 7-22](#) lists the *Extended Status* word in the last descriptor of a packet (*EOP* bit is set). [Table 7-23](#) lists the *Extended Status* word in any descriptor but the last one of a packet (*EOP* bit is cleared).

Table 7-22. Receive Status (RDESC.STATUS) Layout of Last Descriptor

19	18	17	16	15	14	13	12	11	10
BMC	LB	Rsv	TS	TSIP	Rsv			Rsv	UDPV
9	8	7	6	5	4	3	2	1	0

Table 7-23. Receive Status (RDESC.STATUS) Layout of Non-Last Descriptor

19	4	3:2	1	0
Reserved		Rsv	EOP = 0b	DD

Rsv (14:11)

Reserved at zero.

FLM(2)

Flow director filter match indication is set for packets that match these filters.

VP(3), PIF (7)

These bits are described in the legacy descriptor format in [Section 7.1.5](#). The VP bit is not set even if a VLAN tag is present if the PFQDE.HIDE_VLAN bit is set for the queue.

EOP (1) and DD (0)

End of Packet and Done bits are listed in the following table:

DD	EOP	Description
0	X	Software setting of the descriptor when it hands it to hardware.
1	0	A completion status indication for a non last descriptor of a packet that spans across multiple descriptors. In a single packet case the DD bit indicates that the hardware is done with the descriptor and its buffers. Only the <i>Length</i> fields are valid on this descriptor.
1	1	A completion status indication of the entire packet and software might take ownership of its descriptors. All fields in the descriptor are valid (reported by the hardware).

UDPCS (4), L4I (5)

The UDPCS (UDP checksum) is set (together with L4CS bit) when hardware provides UDP checksum offload. The L4I (L4 Integrity) is set when hardware provides any L4 offload as: UDP checksum, TCP checksum or SCTP CRC offload.

IPCS(6)

IPCS as described in Legacy Rx descriptor (in [Section 7.1.5](#)).

OUTERIPCS(8)

Indicates that a checksum was done on the outer IP header of an NVGRE or VXLAN packet.

VEXT (9)

Outer-VLAN is found on a double VLAN packet. This bit is valid only when CTRL_EXT.EXTENDED_VLAN is set. See more details in [Section 7.4.5](#).

UDPV (10)

The *UDP Checksum Valid* bit indicates that the incoming packet contains a valid (non-zero value) checksum field in an incoming fragmented (non-tunneled) UDP IPv4 packet. It means that the *Fragment Checksum* field in the receive descriptor contains the UDP checksum as described in [Section 7.1.6.5](#). When this field is cleared in the first fragment that contains the UDP header, it means that the packet does not contain a valid UDP checksum and the checksum field in the Rx descriptor should be ignored. This field is always cleared in incoming fragments that do not contain the UDP header.

TSIP (15)

Timestamp in packet. The *Timestamp In Packet* bit is set to indicate that the received packet arrival time was captured by the hardware and the timestamp was placed in the receive buffer. For more details see [Section 7.7](#) and [Section 7.1.6.2](#).

TS (16)

The *Time Stamp* bit is set when the device recognized a time sync packet. In such a case the hardware captures its arrival time and stores it in the Time Stamp register. For more details see [Section 7.7](#).

LB (18)

This bit provides a loopback status indication which means that this packet is sent by a local VM (VM to VM switch indication).

BMC (19)

Packet received from BMC. The *BMC* bit is set to indicate the packet was sent by the local BMC. Bit is cleared if packet arrives from the network. For more details see [Section 10.5.2.2](#).

Extended Error (12-bit offset 21, 2nd line)

[Table 7-24](#) and the following text describe the possible errors reported by hardware.

FDIRERR (2:0)

This field is relevant when the flow director filters are enabled.

FDIRErr(0) - Length — If the flow director filter matches the *Length* bit, this indicates that the distance of the matched filter from the hash table exceeds the FDIRCTRL.Max-Length. If there is no matched filter, the *Length* bit is set if the flow director linked list of the matched hash value exceeds the FDIRCTRL.Max-Length.

FDIRErr(1) - Drop — The *Drop* bit indicates that a received packet matched a flow director filter with a drop action. In the case of perfect mode filtering, it is expected to find the drop indication only when the linked list in the flow director bucket exceeds the permitted Max-Length. In this case, the packet is not dropped. Instead, it is posted to the Rx queue (indicated in the filter context) for software handling of the Max-Length exception. In the case of hash mode filtering, it is expected that the drop queue is always a valid queue so all packets that match the drop filter are visible to software.

FDIRErr(2) - Coll — A matched flow director filter with a collision indication was found. The collision indicates that software attempted to step over this filter with a different action that was already programmed.

HBO (3)

The *Header Buffer Overflow* bit is set if the packet header (calculated by hardware) is bigger than the header buffer (defined by SSRCTL.BSIZEHEADER). *HBO* reporting might be used by software to allocate bigger buffers for the headers. It is meaningful only if the *SPH* bit in the receive descriptor is set as well. The *HDR_LEN* field is valid even when the *HBO* bit is set. Packets with HBO error are posted to host memory regardless of the store bad packet setting (FCTRL.SBP). Packet DMA to its buffers when the *HBO* bit is set, depends on the device settings as follows:

SRRCTL.DESCTYPE	DMA Functionality
Header Split (010b)	The header is posted with the rest of the packet data to the packet buffer.
Always Split Mode (101b)	The header buffer is used as part of the data buffers and contains the first SRRCTL.BSIZEHEADER bytes of the packet.

Rsv (5:4)

Reserved at zero.

OUTERIPER (6)

Indicates an error was found in the checksum of an outer IP header of a VXLAN or NVGRE packet or that the UDP checksum of the outer UDP header in a VXLAN packet was not zero.

RXE (9)

RXE is described in the legacy descriptor format in [Section 7.1.5](#).

L4E (10)

L4 integrity error is valid only when the *L4I* bit in the *Status* field is set. It is active if L4 processing fails (TCP checksum or UDP checksum or SCTP CRC). Packets with L4 integrity error are posted to host memory regardless of the store bad packet setting (*FCTRL.SBP*). In case of VXLAN or NVGRE packets, this relates to the internal L4 header.

IPE(11)

IPv4 checksum error is described in [Section 7.1.6](#). In case of VXLAN or NVGRE packets, this relates to the internal IP header.

PKT_LEN (16-bit offset 32, 2nd line)

PKT_LEN holds the number of bytes posted to the packet buffer. The length covers the data written to a receive buffer including posted CRC bytes (if any). Software must read multiple descriptors to determine the complete length for packets that span multiple receive buffers. If *SRRCTL.DESCTYPE* = 2 (advanced descriptor header splitting) and the buffer is not split because the header is bigger than the allocated header buffer, this field reflects the size of the data written to the data buffer (header + data).

When short packets are padded, the *PKT_LEN* does not include the padding, so that the software device driver can detect the location of a potential time stamp in the packet.

When a time stamp is added (*TSIP* bit is set), the *PKT_LEN* includes the timestamp size (+8).

VLAN Tag (16-bit offset 48, 2nd line)

This field is described in the legacy descriptor format in [Section 7.1.5](#). The VLAN tag is set to 0x0000 even if a VLAN tag is present if the *PFQDE.HIDE_VLAN* bit is set for the queue.

7.1.5.3 Receive Descriptor Fetching

The E610 implements a fetch-by-demand mechanism for descriptor fetch. Descriptors are not fetched in advance, but rather fetched after a packet is received. Such a strategy eliminates the need to store descriptors on-die for each and every descriptor queue in anticipation for packet arrival.

7.1.5.4 Receive Descriptor Write-Back

The E610 writes back the receive descriptor immediately following the packet write into system memory. It is therefore possible for a single descriptor to be written at a time into memory. However, if aggregation occurs during descriptor fetch (see [Section 7.1.5.3](#)), the descriptors fetched in the aggregated operation are written back in a single write-back operation.

Note: Software can determine if a packet has been received only by checking the receive descriptor *DD* bit in memory. Checking through *DD* bits (and not by checking the receive head pointer in *RDH/RDL* registers) eliminates a potential race condition: all descriptor data is posted internally prior to incrementing the head register and a read of the head register could potentially pass the descriptor waiting inside the E610.

7.1.5.5 Receive Descriptor Queue Structure

Figure 7-10 shows the structure of each of the receive descriptor rings. Note that each ring uses a contiguous memory space.

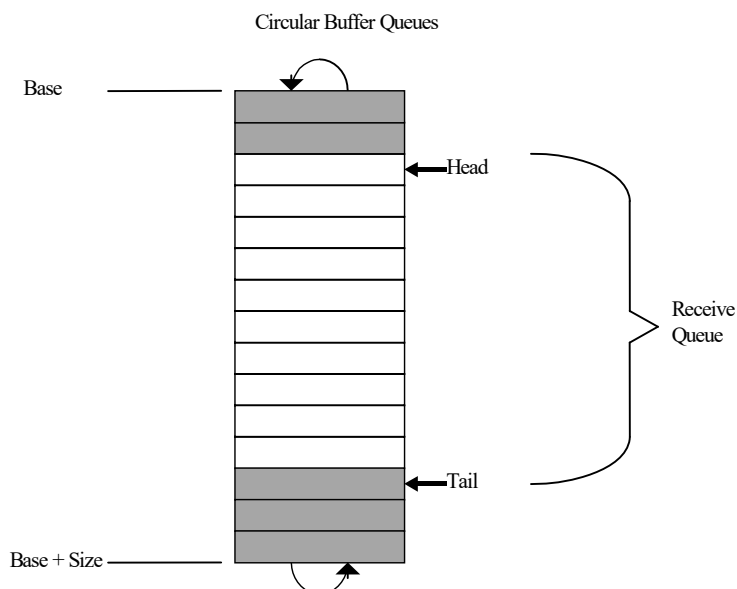


Figure 7-10. Receive Descriptor Ring Structure

Software inserts receive descriptors by advancing the tail pointer(s) to refer to the address of the entry just beyond the last valid descriptor. This is accomplished by writing the descriptor tail register(s) with the offset of the entry beyond the last valid descriptor. The E610 adjusts its internal tail pointer(s) accordingly. As packets arrive, they are stored in memory and the internal head pointer(s) is increased by the E610.

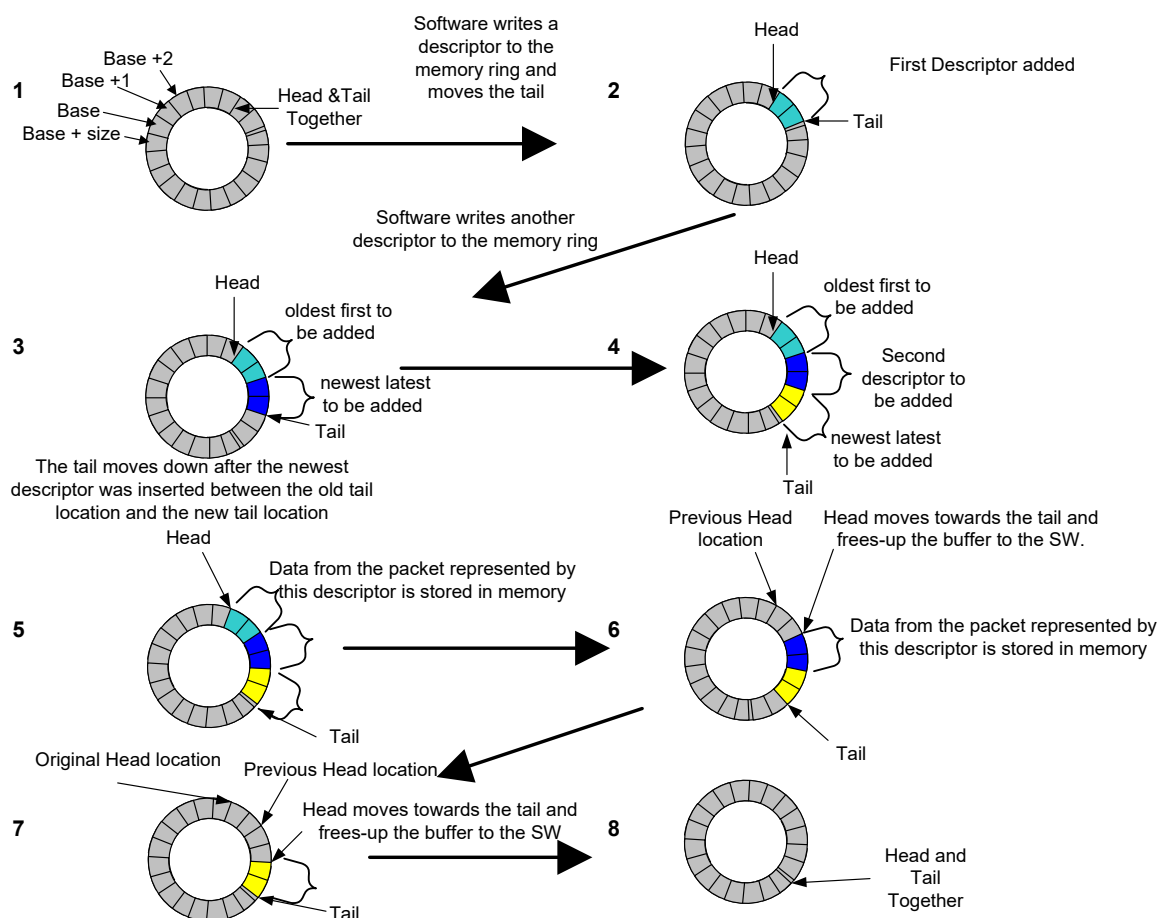


Figure 7-11. Descriptors and Memory Rings

The E610 writes back used descriptors just prior to advancing the head pointer(s). Head and tail pointers wrap back to base when the number of descriptors corresponding to the size of the descriptor ring have been processed.

The receive descriptor head and tail pointers reference to 16-byte blocks of memory. Shaded boxes in Figure 7-11 represent descriptors that have stored incoming packets but have not yet been recognized by software. Software can determine if a receive buffer is valid by reading descriptors in memory rather than by I/O reads. Any descriptor with a *DD* bit set has been used by the hardware, and is ready to be processed by software.

Note: The head pointer points to the next descriptor that is to be written back. At the completion of the descriptor write-back operation, this pointer is increased by the number of descriptors written back. Hardware owns all descriptors between [head... tail]. Any descriptor not in this range is owned by software.

The receive descriptor rings are described by the following registers:

- **Receive Descriptor Base Address registers (RDBA)** — This register indicates the start of the descriptor ring buffer; this 64-bit address is aligned on a 128-byte boundary and is stored in two consecutive 32-bit registers. Hardware ignores the lower 7 bits.

- **Receive Descriptor Length registers (RDLEN)** — This register determines the number of bytes allocated to the circular buffer. This value must be a multiple of 128 (the maximum cache line size). Since each descriptor is 16 bytes in length, the total number of receive descriptors is always a multiple of 8.
- **Receive Descriptor Head registers (RDH)** — This register holds a value that is an offset from the base, and indicates the in-progress descriptor. There can be up to 64K-8 descriptors in the circular buffer. Hardware maintains a shadow copy that includes those descriptors completed but not yet stored in memory.

Software can determine if a packet has been received by either of two methods: reading the *DD* bit in the receive descriptor field or by performing a Programmed I/O read of the Receive Descriptor Head register. Checking the descriptor *DD* bit in memory eliminates a potential race condition. All descriptor data is written to the I/O bus prior to incrementing the head register, but a read of the head register could pass the data write in systems performing I/O write buffering. Updates to receive descriptors use the same I/O write path and follow all data writes. Consequently, they are not subject to the race.

- **Receive Descriptor Tail registers (RDT)** — This register holds a value that is an offset from the base, and identifies the location beyond the last descriptor hardware can process. This is the location where software writes the first new descriptor.

If software statically allocates buffers, and uses a memory read to check for completed descriptors, it simply has to zero the status byte in the descriptor to make it ready for re-use by hardware. This is not a hardware requirement, but is necessary for performing an in-memory scan. This is relevant only to legacy descriptors.

All the registers controlling the descriptor rings behavior should be set before receive is enabled, apart from the tail registers which are used during the regular flow of data.

7.1.5.5.1 Low Receive Descriptors Threshold

As previously described, the size of the receive queues is measured by the number of receive descriptors. During run time, software processes descriptors and upon completion of descriptors, increments the Receive Descriptor Tail registers. At the same time, the hardware may post new received packets incrementing the Receive Descriptor Head registers for each used descriptor.

The number of usable (free) descriptors for the hardware is the distance between the Tail and Head registers. When the tail reaches the head, there are no free descriptors and further packets might be either dropped or block the receive FIFO. To avoid this situation, the E610 might generate a low latency interrupt (associated to the relevant Rx queue) once there are less equal free descriptors than specified by a low level threshold. The threshold is defined in 64 descriptors granularity per queue in the `SRRCTL[n].RDMTS` field.

7.1.6 Receive Offloads

7.1.6.1 Header Splitting

7.1.6.1.1 Purpose

This feature consists of splitting a packet header to a different memory space. This helps the host to fetch headers only for processing: headers are posted through a regular snoop transaction to be processed by the host CPU. It is recommended to perform this transaction with TPH enabled (see [Section 7.5](#)).

The E610's support for header split is controlled by the *DESCTYPE* field of the Split Receive Control registers (SRRCTL). The following modes exist in both split and non-split modes:

- 000b: Legacy mode — Legacy descriptors are used, headers and payloads are not split.
- 001b: Advanced mode, no split — Advanced descriptors are in use, header and payload are not split.
- 010b: Advanced mode, header split — Advanced descriptors are in use, header and payload are split to different buffers.
- 101b: Advanced mode, split always — Advanced descriptors are in use, header and payload are split to different buffers. If no split is done, the first part of the packet is stored in the header buffer.

The E610 uses packet splitting when the SRRCTL[n].*DESCTYPE* is greater than one.

7.1.6.1.2 Description

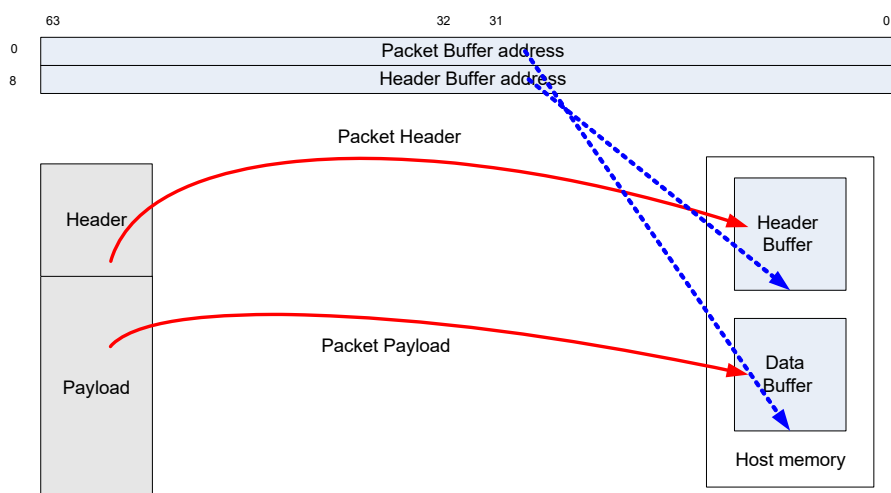


Figure 7-12. Header Splitting Diagram

The physical address of each buffer is written in the *Buffer Addresses* fields:

- The packet buffer address includes the address of the buffer assigned to the packet data.
- The header buffer address includes the address of the buffer that contains the header information. The receive DMA module stores the header portion of the received packets into this buffer.

The sizes of these buffers are statically defined in the SRRCTL[n] registers:

- The *BSIZEPACKET* field defines the size of the buffer for the received packet.
- The *BSIZEHEADER* field defines the size of the buffer for the received header. If header split is enabled, this field must be configured to a non-zero value. The E610 only writes the header portion into the header buffer. The header size is determined by the options enabled in the PSRTYPE registers.

When header split is selected, the packet is split only on selected types of packets. A bit exists for each option in PSRTYPE[n] registers, so several options can be used in conjunction. If one or more bits are set, the splitting is performed for the corresponding packet type. See [Section 8.2.2.8.4](#) for details on the possible header types supported. In virtualization mode, a separate PSRTYPE register is provided per pool up to the number of pools enabled. In non-virtualization mode, only PSRTYPE[0] is used.

Rules regarding header split:

- Packets that have headers bigger than 1023 bytes are not split.
- The header of a fragmented IPv6 packet is defined until the *Fragment* extension header.
- An IP in IP packet (such as any combination of IPv4 and IPv6 tunneling) is not split. Not relevant for NVGRE and VXLAN packets
- Packet header cannot span across buffers, therefore, the size of the header buffer must be larger than any expected header size. In case of header split mode (SRRCTL.DESCTYPE = 010b), a packet with a header larger than the header buffer is not split.

Table 7-25 lists the behavior of the E610 in the different modes.

Table 7-25. Behavior in Header Split Modes

DESCTYPE	Condition	SPH	HBO ¹	PKT_LEN ²	HDR_LEN ²	Header and Payload DMA ³
Split	1. Header can't be decoded	0	0	Min (packet length, buffer size)	0x0	Header + Payload (+ TimeStamp) (+ Padding) -> Packet Buffer
	2. Header <= BSIZEHEADER and Payload > 0	1	0	Min (payload length, buffer size)	Header size	Header -> Header Buffer Payload (+ TimeStamp) (+ Padding) -> Packet Buffer
	3. Header <= BSIZEHEADER and Payload = 0 (Header only packet)	1	0	0	Header size	Header (+ TimeStamp) (+ Padding) -> Header Buffer ⁴
	4. Header > BSIZEHEADER	1	1	Min (packet length, buffer size)	Header size ⁵	Header + Payload (+ TimeStamp) -> Packet Buffer
Split – always use header buffer	1. Header can't be decoded and packet length <= BSIZEHEADER	0	0	0x0	Packet length	Header + Payload (+ TimeStamp) (+ Padding) -> Header Buffer
	2. Header cannot be decoded and packet length > BSIZEHEADER	0	0	Min (packet length – BSIZEHEADER, data buffer size)	BSIZEHEADER	Header + Payload (+ TimeStamp) -> Header + Packet Buffers ⁵
	3. Header <= BSIZEHEADER	1	0	Min (payload length, data buffer size)	Header Size	Header -> Header Buffer Payload (+ TimeStamp) (+ Padding) -> Packet Buffer
	4. Header > BSIZEHEADER	1	1	Min (packet length – BSIZEHEADER, data buffer size)	Header Size ⁶	Header + Payload (+ TimeStamp) -> Header + Packet Buffer

1. HBO is set to 1b if the header size is bigger than BSIZEHEADER and zero otherwise.

2. PKT_LEN and HDR_LEN includes optionally added time stamp if TSIP bit is set and does not include padding added by the device.

3. Partial means up to BSIZEHEADER.

4. Even if there is no payload at all and there is a timestamp in the packet - the timestamp is considered as an 8 bytes payload and is written to the packet buffer.

5. If the packet spans more than one descriptor, only the header buffer of the first descriptor is used.

6. HDR_LEN does not reflect the actual data size stored in the header buffer. It reflects the header size determined by the parser.

7.1.6.2 Receive Packet Timestamp in Buffer

The E610 supports adding an optional tailored header appended to the end of the packet in the receive buffer. The tailored header includes a 64-bit timestamp composed of the packet reception time measured in the SYSTIMEL (Low DW) and SYSTIMEH (High DW) registers (See [Section 7.7.3.2](#) for further information on SYSTIMEL/H operation). The timestamp starts right at the end of the packet (after the last byte). It is sent as {SYSTIMEH, SYSTIMEL} - most significant byte first (closest to packet).

PKT_LEN and *HDR_LEN* includes optionally added time stamp.

When the *TSAUXC.DISABLE_SYSTIME* bit is cleared and the *TSYNCRXCTL.TSIP_UP_EN* is set for the UP in the packet (or *TSYNCRXCTL.TSIP_UT_EN* for un-tagged packets), packets received that meet the *TSYNCRXCTL.TYPE* are time-stamped. A packet that was time stamped is reported as follows:

- Place a 64-bit timestamp, indicating the time a packet was received by the MAC, appended at the end of the received packet within the receive buffer.
- Set the *TSIP* bit in the *RDESC.STATUS* field of the last receive descriptor.

When packets are coalesced, the timestamp reflects the reception time of the last coalesced fragment, unless the packet is a pure ACK packet, in which case, the timestamp is of the first packet.

7.1.6.3 Receive Checksum Offloading

The E610 supports the offloading of three receive checksum calculations: the fragment checksum, the IPv4 header checksum, and the TCP/UDP checksum.

For supported packet/frame types, the entire checksum calculation can be offloaded to the E610. The E610 calculates the IPv4 checksum and indicates a pass/fail indication to software via the *IPv4 Checksum Error* bit (*RDESC.IPE*) in the *ERROR* field of the receive descriptor. Similarly, the E610 calculates the TCP or UDP checksum and indicates a pass/fail condition to software via the *TCP/UDP Checksum Error* bit (*RDESC.TCPE*). For NVGRE or VXLAN packets, the *IPE* and *TCPE* bits relates to the inner IP/TCP header. The outer IPv4 checksum is also checked and a pass/fail indication is indicated to software via the *Outer IPv4 Checksum Error* bit (*OUTERIPER*). These error bits are valid when the respective status bits indicate the checksum was calculated for the packet (*RDESC.IPCS*, *RDESC.L4CS*, and *RDESC.OUTERIPCS* respectively).

Similarly, if *RFCTL.IPV6_DIS* and *RFCTL.IP6XSUM_DIS* are cleared to zero, the E610 calculates the TCP or UDP checksum for IPv6 packets. It then indicates a pass/fail condition in the *TCP/UDP Checksum Error* bit (*RDESC.TCPE*).

Note: Only Ethernet II frames are supported (no checksum support for SNAP packets).

Table 7-26. Supported Receive Checksum Capabilities

Packet Type	Hardware IP Checksum Calculation	Hardware TCP/UDP Checksum Calculation
IP header's protocol field contains a protocol # other than TCP or UDP.	Yes	No
IPv4 + TCP/UDP packets.	Yes	Yes
IPv6 + TCP/UDP packets.	No (N/A)	Yes
IPv4 packet has IP options (IP header is longer than 20 bytes).	Yes	Yes

Table 7-26. Supported Receive Checksum Capabilities [continued]

Packet Type	Hardware IP Checksum Calculation	Hardware TCP/UDP Checksum Calculation
IPv6 packet with next header options: - Hop-by-hop options. - Destinations options (without home address). - Destinations options (with home address). - Routing (with segment left 0). - Routing (with segment left > 0). - Fragment.	No (N/A) No (N/A) No (N/A) No (N/A) No (N/A) No (N/A)	Yes Yes No Yes No No
Packet has TCP or UDP options.	Yes	Yes
IPv4 tunnels: - IPv4 packet in an IPv4 tunnel. - IPv6 packet in an IPv4 tunnel.	Yes (outer IPv4 only) Yes (IPv4)	No No
IPv6 tunnels: - IPv4 packet in an IPv6 tunnel. - IPv6 packet in an IPv6 tunnel.	No No (N/A)	No No
Packet is an IPv4 fragment.	Yes	UDP checksum assist
Packet is greater than 1522 bytes.	Yes	Yes
Packet has 802.3ac tag.	Yes	Yes
NVGRE: - Inner IPv4 packet - Inner IPv6 packet	Yes (Inner and outer) Yes (outer)	Yes Yes
VXLAN: - Inner IPv4 packet - Inner IPv6 packet	Yes (Inner and outer) Yes (outer)	Yes (inner only) ¹ Yes (inner only)

1. The outer UDP header of VXLAN packets do not use a checksum.

7.1.6.4 SCTP Receive Offload

If a receive packet is identified as SCTP, the E610 checks the CRC32 checksum of this packet and identifies this packet as SCTP. Software is notified of the CRC check via the *L4I* and *L4E* bits in the *Extended Status* field and *Extended Error* field in the Rx descriptor. The detection of an SCTP packet is indicated via the *SCTP* bit in the *Packet Type* field of the Rx descriptor. SCTP CRC uses the CRC32c polynomial as follows (0x11EDC6F41):

$$X_{32}+X_{28}+X_{27}+X_{26}+X_{25}+X_{23}+X_{22}+X_{20}+X_{19}+X_{18}+X_{14}+X_{13}+X_{11}+X_{10}+X_9+X_8+X_6+X_0$$

The checker assumes the following SCTP packet format.

Table 7-27. SCTP Header

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																
Source Port																Destination Port																															
Verification Tag																																															
CRC Checksum (CRC32c)																																															
Chunks 1..n																																															

7.1.6.5 Receive UDP Fragmentation Checksum

The E610 might provide a receive fragmented UDP checksum offload for IPv4 non-tunneled packets. The RXCSUM.PCSD bit should be cleared and the RXCSUM.IPPCSE bit should be set to enable this mode.

The following table lists the outcome descriptor fields for the following incoming packets types.

Incoming Packet Type	Fragment Checksum	UDPV	UDPCS / L4CS
Non-IP packet	0	0	0
IPv6 packet	0	0	Depends on transport UDP: 1 / 1 TCP: 0 / 1
Non fragmented IPv4 packet			
Fragmented IPv4 with protocol = UDP, first fragment (UDP protocol present)	The unadjusted 1's complement checksum of the IP payload if Checksum in packet header is different than zero and zero otherwise.	1 if the UDP header checksum is valid (not 0)	0 / 0
Fragmented IPv4, when not first fragment	The unadjusted 1's complement checksum of the IP payload	1	0 / 0
Fragmented IPv4 with protocol = UDP, first fragment (UDP protocol present) within a VXLAN or NVGRE packet.	The unadjusted 1's complement checksum of the inner IP payload	1 if the UDP header checksum is valid (not 0)	1 / 0
Fragmented IPv4 when not first fragment within a VXLAN or NVGRE packet.	The unadjusted 1's complement checksum of the inner IP payload	0	1 / 0

Note: When the driver computes the 16-bit ones complement sum on the incoming packets of the UDP fragments, it should expect a value of 0xFFFF.

7.1.7 Receive Statistics

7.1.7.1 General Rules

- All Statistics registers are cleared on read. In addition, they stick at 0xFF...F when the maximum value is reached.
- For the receive statistics it should be noted that a packet is indicated as received if it passes the device filters and is placed into the packet buffer memory. A packet does not have to be DMA'ed to host memory to be counted as received.
- Due to divergent paths between interrupt-generation and logging of relevant statistics counts, it might be possible to generate an interrupt to the system for a noteworthy event prior to the associated statistics count actually being increased. This is extremely unlikely due to expected delays associated with the system interrupt-collection and ISR delay, but might be an explanation for interrupt statistics values that do not quite make sense. Hardware guarantees that any event noteworthy of inclusion in a statistics count is reflected in the appropriate count within 1 μ s; a small time-delay prior to reading the statistics might be required to avoid a potential mismatch between and interrupt and its cause.
- All byte (octet) counters composed of 2 registers can be fetched by two consecutive 32-bit accesses while reading the Low 32-bit register first or a single 64-bit access.

- All receive statistic counters in the Filter unit (listed below) might count packets that might be dropped by the packet buffer or receive DMA. Same comment is valid for the byte counters associated with these packet counters: PRC64, PRC127, PRC255, PRC511, PRC1023, PRC1522, BPRC, MPRC, GPRC, RXNFGPC, RUC, ROC.

7.1.7.2 Receive Statistics Hierarchy

The following diagram describes the relations between the packet flow and the different statistic counters.

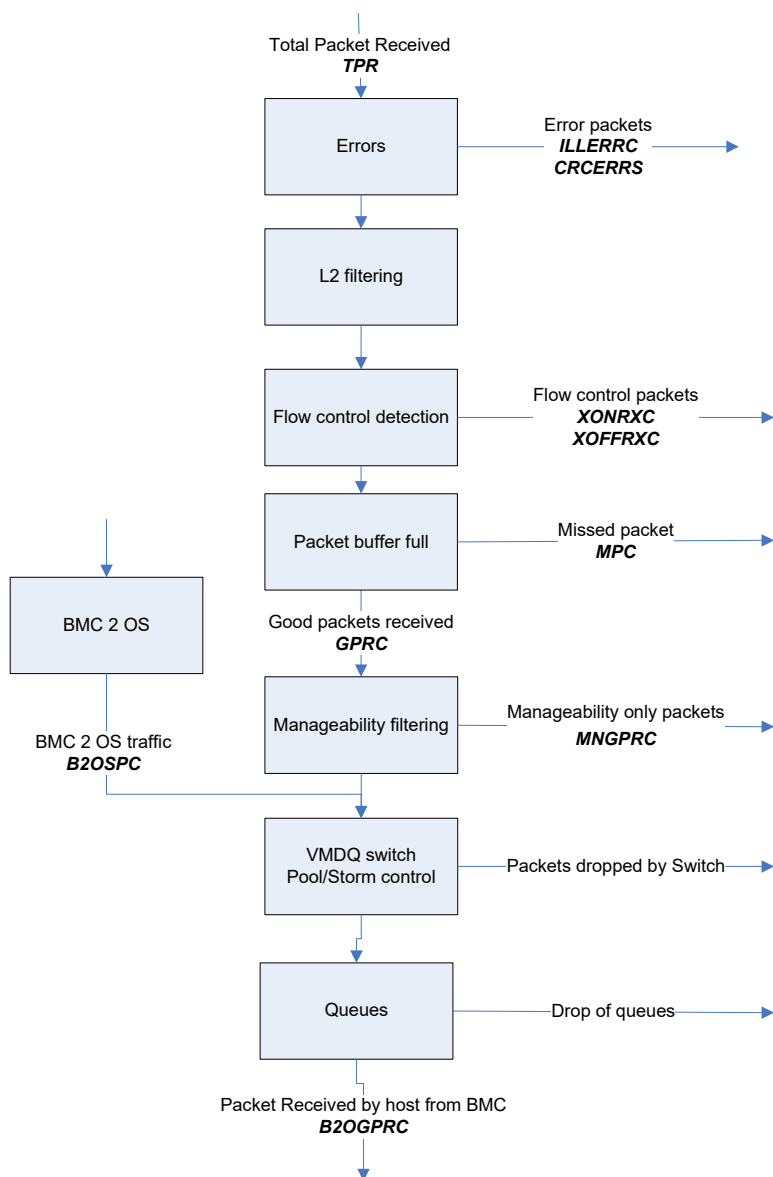


Figure 7-13. Receive Flow Statistics

7.2 Transmit Functionality

7.2.1 Packet Transmission

Transmit packets are made up of data buffers in host memory that are indicated to hardware by pointer and length pairs. These pointer and length pairs are named as transmit descriptors that are stored in host memory as well.

Software prepares memory structures for transmission by assembling a list of descriptors. It then indicates this list to hardware for updating the on-chip transmit tail pointer. Hardware transmits the packet only after it has completely fetched all packet data from host memory and deposited it into the on-chip transmit FIFO. This store and forward scheme enables hardware-based offloads such as TCP or UDP checksum computation, and many other ones detailed in this document while avoiding any potential PCIe under-runs.

7.2.1.1 Transmit Storage in System Memory

A packet (or multiple packets in transmit segmentation) can be composed of one or multiple buffers. Each buffer is indicated by a descriptor. Descriptors of a single packet are consecutive, while the first one points to the first buffer and the last one points to the last buffer (see [Figure 7-14](#)). The following rules must be kept:

- Address alignment of the data buffers can be on any byte boundary.
- Data buffers of any transmitted packet must include at least the 12 bytes of the source and destination Ethernet MAC Addresses as well as the 2 bytes of the *Type/Len* field.
- A packet (or multiple packets in transmit segmentation) can span any number of buffers (and their descriptors) up to a limit of 40 minus *WTHRESH* minus 2 (see [Section 7.2.3.3](#) for Tx Ring details and [Section 7.2.3.5.1](#) for *WTHRESH* details). For best performance it is recommended to minimize the number of buffers as possible.

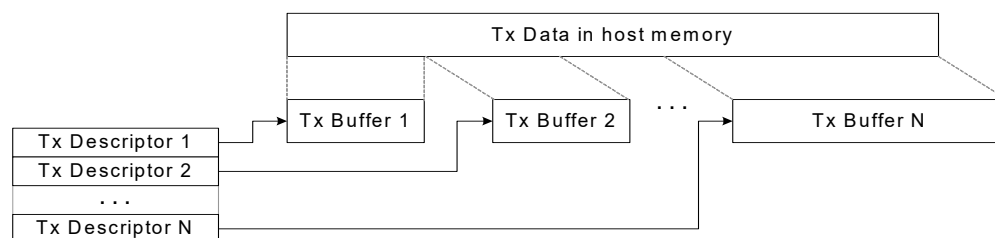


Figure 7-14. Tx Packet in Host Memory

7.2.1.2 Transmit Path in the E610

The transmit path in the E610 consists of the following stages:

- Descriptor plane
 - The E610 maintains a set of 128 on-die descriptor queues. Each queue is associated with a single descriptor ring in system memory. See [Section 7.2.3.3](#) for more details on the Tx descriptor rings. Each on-die descriptor queue contains up to 40 descriptors to achieve the desired performance.
 - A fetch mechanism loads Tx descriptors from the descriptor rings in system memory to the respective descriptor queues in the E610. A descriptor fetch arbiter determines the order in which descriptors are fetched into the various on-die descriptor queues. See [Section 7.2.3.4](#) for more details on the fetch mechanism.
 - An arbitration scheme determines the order in which descriptors are processed and requests are generated for data reads. These requests load packet data from system memory into a set of Tx packet buffers. The arbitration mechanism varies with configuration and is described in [Section 7.8](#).
 - Once a packet has been fetched into a packet buffer, status is (optionally) written back into system memory. See [Section 7.2.3.5](#) for more details.
- Packet plane (data plane)
 - Packet data is stored in up to eight packet buffers. The number and size of packet buffers vary with the mode of operation and is described in [Section 7.2.1.2.2](#).
 - If more than a single packet buffer is enabled, an arbitration scheme determines the order in which packets are taken out of the packet buffers and sent to the MAC for transmission. The arbitration mechanism is described in [Section 7.8](#).

7.2.1.2.1 Tx Queues Assignment

The E610 supports a total of 128 queues per LAN port. Each Tx queue is associated with a packet buffer and the association varies with the operational mode. The following mechanisms impact the association of the Tx queues. These are described briefly in this section, and in full details in separate sections:

- **Virtualization (VT)** — In a virtualized environment, DMA resources are shared between more than one software entity (operating system and/or device driver). This is done through allocation of transmit descriptor queues to virtual partitions (VMM, IOVM, VMs, or VFs). Allocation of queues to virtual partitions is done in sets of queues of the same size, called queue pools, or **pools**. A pool is associated with a single virtual partition. Different queues in a pool can be associated with different packet buffers. For example, in a DCB system, each of the queues in a pool might belong to a different TC and therefore to a different packet buffer. The PFVFTE register contains a bit per VF. When the bit is set to 0b, packet transmission from the VF is disabled. When set to 1b, packet transmission from the VF is enabled.
- **DCB** — DCB provides QoS through priority queues, priority flow control, and congestion management. Queues are classified into one of several (up to eight) Traffic Classes (TCs). Each TC is associated with a single unique packet buffer.
- **Transmit fanout** — A single descriptor queue might be enough for a given functionality. For example, in a VT system, a single Tx queue can be allocated per VM. However, it is often the case that the data rate achieved through a single buffer is limited. This is especially true with 10 GbE, and traffic needs to be divided into several Tx queues to reach the desired data rate. Therefore, multiple queues might be provided for the same functionality.

Table 7-28 lists the queuing schemes. Scheme selection is done via the MTQC register.

Table 7-28. Tx Queuing Schemes

VT	DCB	Queues Allocation	Packet Buffers Allocation
No	No	A single set of 64 queues is assigned to a single packet buffer. Queues 64...127 should not be used.	A single packet buffer for all traffic
No	Yes	Eight TCs mode — allocation of 32-32-16-16-8-8-8-8 queues for TC0-TC1-...- TC7, respectively. Four TCs mode — allocation of 64-32-16-16 queues for TC0-TC1-...- TC3, respectively.	A separate packet buffer is allocated to each TC (total of four or eight).
Yes	No	32 pools x 4 queues, or 64 pools x 2 queues	A single packet buffer for all traffic.
Yes	Yes	16 pools x 8 TCs, or 32 pools x 4 TCs	A separate packet buffer is allocated to each TC (total of four or eight).

Note: Software can use any number of queues per each TC or per each pool within the allocated ranges previously described by disabling any unused queue.

Note: Programming MTQC must be done only during the init phase, while software must also set RTTDCS.ARBDIS before configuring MTQC and then clear RTTDCS.ARBDIS afterwards.

Table 7-29. Queues, Pools, and TCs programming

Device Setting (MTQC)			Device Functionality	
DCB_ena	VT_Ena	NUM_TC_OR_Q	Total Number of Tx Queues	TC & Pools
0	0	00	64	-
<> 0	<> 0	00	Non valid configuration	
0	0	<> 00	Non valid configuration	
1	0	01	N/A	
1	0	10	128	TC0 — TC3
1	0	11	128	TC0 — TC7
0	1	01	128	64 pools
0	1	10	128	32 pools
0	1	11	N/A	
1	1	01	N/A	
1	1	10	128	TC0 — TC3 & 32 pools
1	1	11	128	TC0 — TC7 & 16 pools

Allocating descriptor queues to VFs uses a consistent indexing over the different Tx queuing schemes. The most significant bits of the queue index represent the VF index, and the least significant bits are either the TC index or are used by software to dispatch traffic according to a Transmit Side Scaling (TSS) algorithm — similar to RSS in the Rx path.

The Tx queue numbers associated with the TCs are listed in the following tables: [Table 7-30](#) and [Table 7-31](#).

Table 7-30. Tx Queues Indexing When VT-on

VT Mode	Allocation of Queue Index Bits According to						
	6	5	4	3	2	1	0
64 VFs + TSS	VF or pool (63..0)						TSS
32 VFs + TSS <i>or</i> 4 TCs	VF or pool (31..0)					TSS / TC	
16 VFs + 8 TCs	VF (15..0)				TC		

Table 7-31. Tx Queues Indexing When VT-off and DCB-on

TC Mode	TCn	# of Qs	Queues Attached to TCn
4 TCs	TC0	64	0xxxxxx
	TC1	32	10xxxxx
	TC2	16	110xxxx
	TC3	16	111xxxx
8 TCs	TC0	32	00xxxxx
	TC1	32	01xxxxx
	TC2	16	100xxxx
	TC3	16	101xxxx
	TC4	8	1100xxx
	TC5	8	1101xxx
	TC6	8	1110xxx
	TC7	8	1111xxx

Note: "x" refers to both 0 or 1, and is used by software to dispatch Tx flows via TSS algorithm.

7.2.1.2.2 Tx Packet Buffers

As previously described, the following modes exist for the E610 packet buffers:

- A single 160 KB packet buffer that serves all Tx descriptor queues, leading to one single (or no) TC enabled, TC0
- Four 40 KB packet buffers, one per enabled TC, leading to four TCs, TC0 to TC3
- Eight 20 KB packet buffers, one per enabled TC, leading to eight TCs, TC0 to TC7

The size of the Tx packet buffer(s) is programmed via the TXPBSIZE registers, one register per TC. Null-sized packet buffer corresponds to a disabled TC.

Note: Setting the packet buffers' size leads to a different partition of a shared internal memory and must be done during boot, prior to communicating, and followed by a software reset.

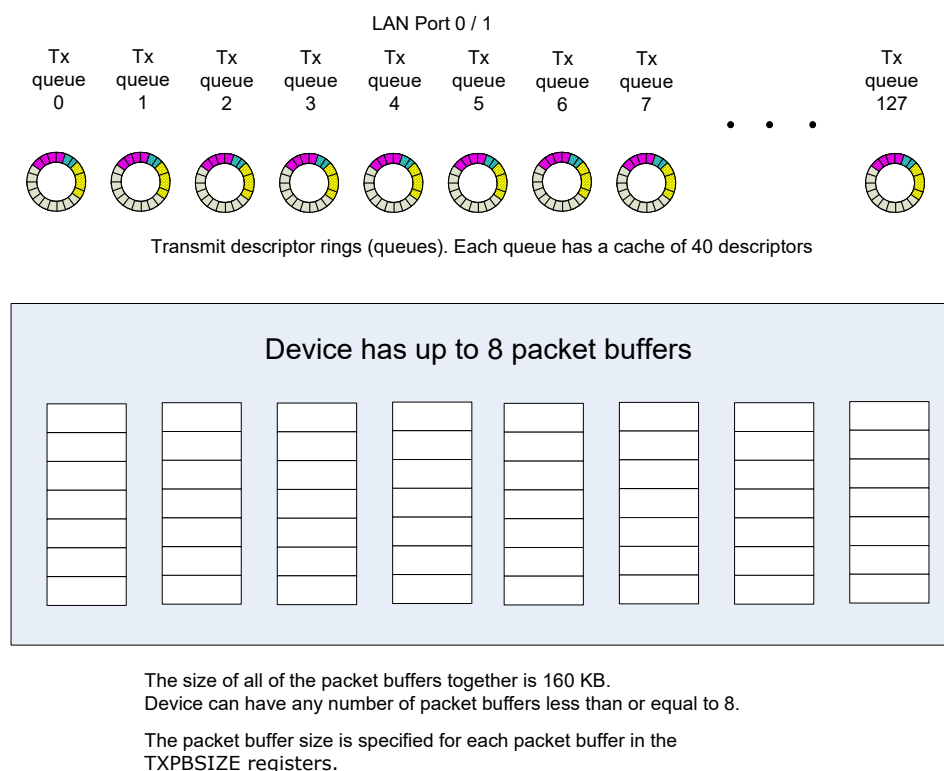


Figure 7-15. Tx Packet Buffers

7.2.1.2.3 Tx Arbitration Schemes

There are basically four Tx arbitration schemes, one per each combination of the DCB and Virtualization (VT) enabled/disabled modes. They are configured via the `MTQC.MTQE` register field.

7.2.1.2.3.1 DCB-on/VT-on

When both DCB and virtualization are enabled, queues are allocated to the packet buffers in a fixed manner, the same number of queues per each TC. Two DCB modes are supported, four TCs or eight TCs mode, according to coherent configuration made in registers `TXPBSIZE` and `MTQC`.

Descriptor Plane Arbiters and Schedulers:

- **Rate-Scheduler** — Once a frame has been fetched out from a rate-limited queue, the next time another frame could be fetched from that queue is regulated by the rate-scheduler. In the meantime, the queue is considered as if it was empty (such as switched-off) for the subsequent arbitration layers.
- **VM Weighted Round Robin Arbiter** — Descriptors are fetched out from queues attached to the same TC in a frame-by-frame weighted round-robin manner, while taking into account any limitation as previously described. Weights or credits allocated to each queue are configured via the `RTTDT1C` register. Bandwidth unused by one queue is reallocated to the other queues within the TC, proportionally to their relative bandwidth shares. TC bandwidth limitation is distributed across all the queues attached to the TC, proportionally to their relative bandwidth shares. Details on weighted round-robin arbiter between the queues can be found in [Section 7.6.2.3](#). It is assumed

traffic is dispatched across the queues attached to a same TC in a straightforward manner, according to the VF to which it belongs.

- **TC Weighted Strict Priority Arbiter** — Descriptors are fetched out from queues attached to different TCs in a frame-by-frame weighted strict-priority manner. Weights or credits allocated to each TC are configured via RTTDT2C registers. Bandwidth unused by one TC is reallocated to the others, proportionally to their relative bandwidth shares. Link bandwidth limitation is distributed across all the TCs, proportionally to their relative bandwidth shares. Details on weighted strict-priority arbiter between the TCs can be found at [Section 7.6.2.3](#). It is assumed (each) driver dispatches traffic across the TCs according to the 802.1p *User Priority* field inserted by the operating system and according to a user priority-to-TC Tx mapping table.

Packet Plane Arbiters:

- **TC Weighted Strict Priority Arbiter** — Packets are fetched out from the different packet buffers in a frame-by-frame weighted strict-priority manner. Weights or credits allocated to each TC (such as to each packet buffer) are configured via RTTPT2C registers, with the same allocation done at the descriptor plane. Bandwidth unused by one TC and link bandwidth limitation is distributed over the TCs as in the descriptor plane. Details on weighted strict-priority arbiter between the TCs can be found in [Section 7.6.2.3](#).
- **Priority Flow Control** packets are inserted with strict priority over any other packets.
- **Manageability** packets are inserted with strict priority over data packets from the same TC, with respect to the bandwidth allocated to the concerned TC. TCs that belong to manageability packets are controlled by MNGTXMAP.MAP.

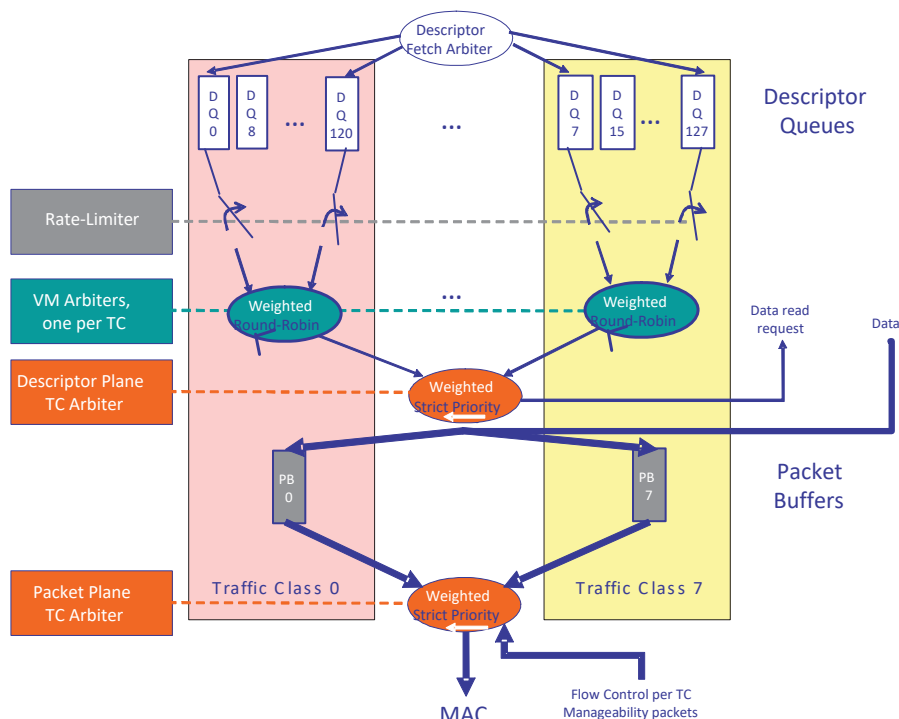


Figure 7-16. Transmit Architecture DCB-on/VT-on — Eight TCs Mode

Note: Replicating TC arbiters before and after the packet buffers is required to provide arbitration whether PCI bandwidth is smaller or greater than the link bandwidth, respectively.

7.2.1.2.3.2 DCB-on/VT-off

When DCB is enabled and virtualization disabled, queues are allocated to the packet buffers in a fixed manner according to the number of TCs. Two DCB modes are supported, four TCs or eight TCs mode, according to coherent configuration made in registers TXPBSIZE and MTQC. In Figure 7-17, eight TCs mode is shown.

- The unique difference with the DCB-on/VT-on arbitration scheme previously described is that the VM weighted round-robin arbiters are degenerated into simple frame-by-frame round-robin arbiters across the queues attached to the same TC. It is assumed driver dispatches traffic across the queues attached to a same TC according to hashing performed on MAC destination addresses. This is aimed to minimize crosstalk between rate-limited and non-rate-limited flows.

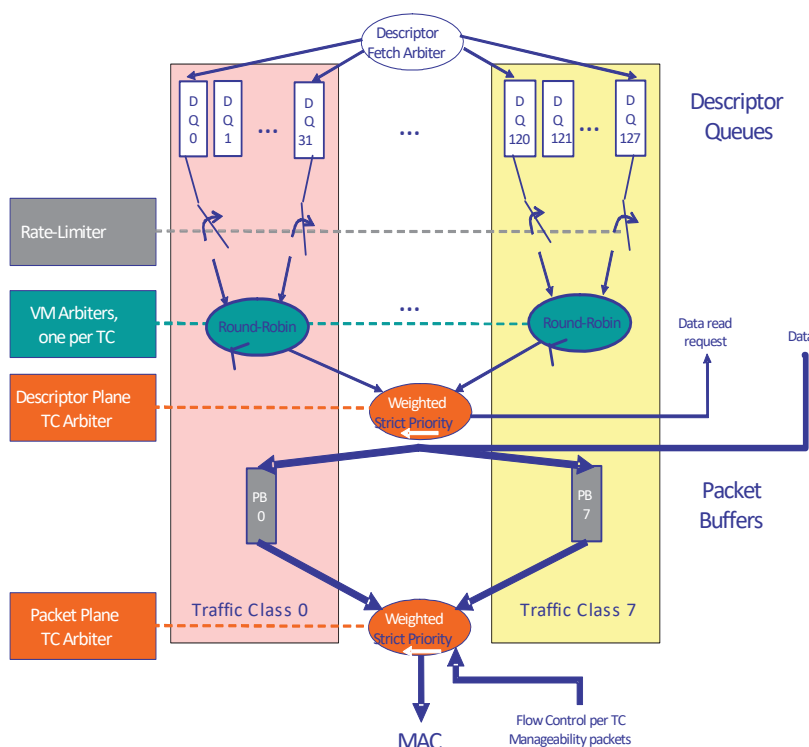


Figure 7-17. Transmit Architecture DCB-on/VT-off — Eight TCs Mode

7.2.1.2.3.3 DCB-off/VT-on

When DCB is disabled and virtualization enabled, all the 128 queues are allocated to a single packet buffer PB(0). Queues are grouped into 32 or 64 pools of 4 or 2 queues, respectively. The number of queue pools corresponds to the number of VFs exposed. Queues are attached to pools according to consecutive indexes

- For the 32 pools case, queues 0, 1, 2, 3 are attached to VF0, queues 4, 5, 6, 7 are attached to VF1, and so forth up to VF31.
- For the 64 pools case, queues 0 and 1 are attached to VF0, queues 2 and 3 are attached to VF1, and so forth up to VF63.

Descriptor Plane Arbiters:

- **Descriptor Queues Round Robin Arbiter** — Descriptors are fetched out from the internal descriptor queues attached to the same pool in a frame-by-frame round-robin manner. It is assumed driver dispatches traffic across the queues of a same pool according to some Transmit Side Scaling (TSS) algorithm similarly to what is done by hardware in the Rx path with RSS.
- **VM Weighted Round Robin Arbiter** — Descriptors are fetched out from queues attached to different pools in a frame-by-frame weighted round-robin manner. Weights or credits allocated to a pool are those allocated for the lowest queue of the pool via the RTTDT1C register. Bandwidth unused by one pool is reallocated to the others proportionally to their relative bandwidth shares. Link bandwidth limitation is distributed across all the pools, proportionally to their relative bandwidth shares. Details on weighted round-robin arbiter between the pools can be found in [Section 7.6.2.3](#).

Packet Plane Arbiter:

- **Manageability** packets are inserted with strict priority over data packets.

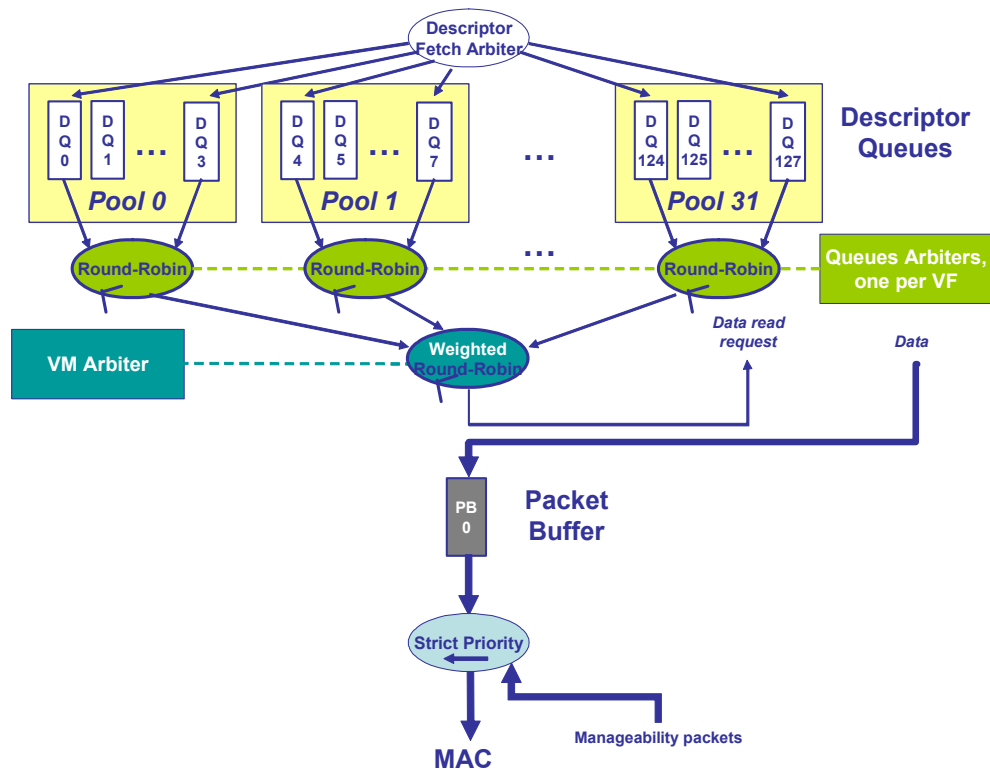


Figure 7-18. Transmit Architecture DCB-off/VT-on — 32 VFs

7.2.1.2.3.4 DCB-off/VT-off

When both DCB and virtualization features are disabled, a single set of up to 64 queues is allocated to a single packet buffer PB(0).

Descriptor Plane Arbiter:

- **Descriptor Queues Round Robin Arbiter** — Descriptors are fetched out from the internal descriptor queues in a frame-by-frame round-robin manner. It is assumed driver dispatches traffic across the queues according to some TSS algorithm similarly to what is done by hardware in the Rx path with RSS.

Packet Plane Arbiter:

- **Manageability** packets are inserted with strict priority over data packets.

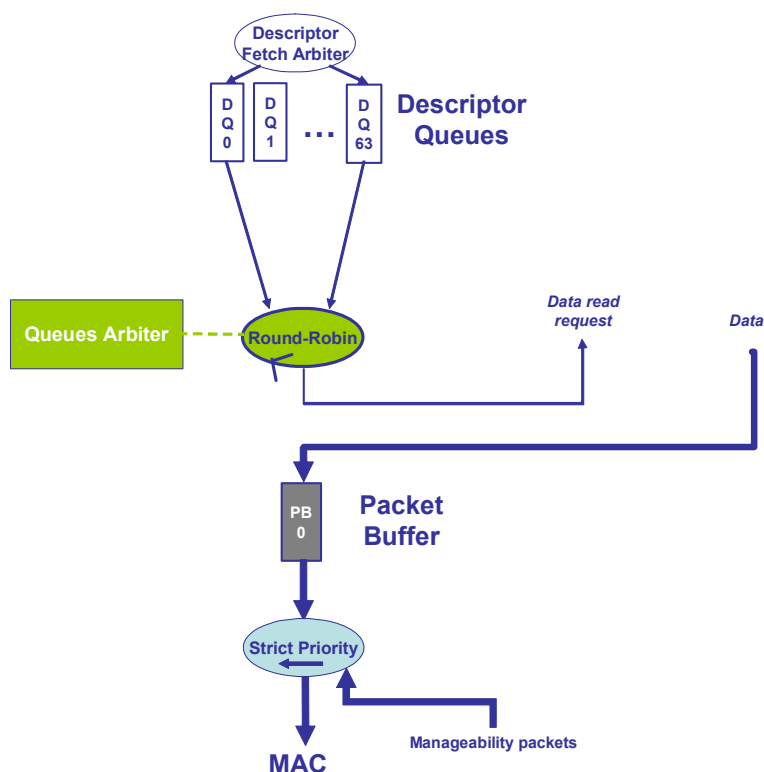


Figure 7-19. Transmit Architecture DCB-off/VT-off

7.2.2 Transmit Contexts

The E610 provides hardware checksum offload and TCP segmentation facilities. These features enable TCP and UDP packet types to be handled more efficiently by performing additional work in hardware, thus reducing the software overhead associated with preparing these packets for transmission. Part of the parameters used to control these features are handled through contexts.

A context refers to a set of parameters providing a particular offload functionality. These parameters are loaded by unique descriptors named transmit context descriptors. A transmit context descriptor is identified by the *DTYP* field (described later in this section) equals to 0x2.

The E610 supports two contexts for each of its 128 transmit queues. The *IDX* bit contains an index to one of these two contexts. Each advanced data descriptor that uses any of the advanced offloading features must refer to a context by the *IDX* field.

Contexts can be initialized with a transmit context descriptor and then used for a series of related transmit data descriptors. Software can use these contexts as long lived ones, while one of the two contexts is used for checksum offload and the other one for transmit segmentation detailed in the following sections. The contexts should be modified when new offload parameters are required.

7.2.3 Transmit Descriptors

7.2.3.1 Introduction

The E610 supports legacy descriptors and advanced descriptors.

Legacy descriptors are intended to support legacy drivers, to enable fast platform power up and to facilitate debug. The legacy descriptors are recognized as such based on *DEXT* bit (see the sections that follow). Legacy descriptors are not supported together with DCB, virtualization. These modes are recognized by a dedicated enable bit for each.

In addition, the E610 supports two types of advanced transmit descriptors:

1. Advanced transmit context descriptor, *DTYP* = 0010b
2. Advanced transmit data descriptor, *DTYP* = 0011b

Note: *DTYP* = 0000b and 0001b are reserved values.

The transmit data descriptor (both legacy and advanced) points to a block of packet data to be transmitted. The advanced transmit context descriptor does not point to packet data. It contains control/context information that is loaded into on-chip registers that affect the processing of packets for transmission. The following sections describe the descriptor formats.

7.2.3.2 Transmit Descriptors Formats

7.2.3.2.1 Notations

This section defines the structure of descriptors that contain fields carried over the network. At the moment, the only relevant field is the *VLAN Tag* field.

The rule for VLAN tag is to use network ordering (also called big endian). It appears in the following manner in the descriptor:

Table 7-32. VLAN Tag

Byte address N + 1 -> first byte on the wire Bit 7			Byte address N -> second byte on the wire Bit 7 -> last on the wire			Bit 0
PRI (3 bits)	DEI	VID (4 bits)				VID (8 bits)

7.2.3.2.2 Legacy Transmit Descriptor Format

To select legacy mode operation, bit 29 (*TDESC.DEXT*) should be set to 0b. In this case, the descriptor format is defined as listed in Table 7-33. Address and length must be supplied by software on all descriptors. Bits in the command byte are optional, as are the *CSO*, and *CSS* fields.

Table 7-33. Transmit Descriptor (TDESC) Layout — Legacy Mode

	63	48	47	40	39	36	35	32	31	24	23	16	15	0
0	Buffer Address [63:0]													
8	VLAN			CSS		Rsvd		STA		CMD		CSO		Length

Table 7-34. Transmit Descriptor Write-Back Format — Legacy Mode

	63	48	47	40	39	36	35	32	31	24	23	16	15	0
0	Reserved								Reserved					
8	VLAN			CSS		Rsvd		STA		CMD		CSO		Length

Buffer Address (64) and Length (16)

The buffer address is a byte-aligned address. Length (*TDESC.LENGTH*) specifies the length in bytes to be fetched from the buffer address provided. The maximum length associated with a single descriptor is 15.5 KB while the total frame size must meet the maximum supported frame size. There is no limitation for the minimum buffer size.

Note: Descriptors with zero length (null descriptors) transfer no data. Null descriptors might appear only between packets and must have their *EOP* bits set.

Checksum Offset and Start — *CSO* (8) and *CSS* (8)

A *Checksum Offset* (*TDESC.CSO*) field indicates where, relative to the start of the packet, to insert a TCP checksum if this mode is enabled. A *Checksum Start* (*TDESC.CSS*) field indicates where to begin computing the checksum. Note that *CSO* and *CSS* are meaningful only in the first descriptor of a packet.

Both *CSO* and *CSS* are in units of bytes. These must both be in the range of data provided to the device in the descriptor. This means for short packets that are padded by software, *CSO* and *CSS* must be in the range of the un-padded data length, not the eventual padded length (64 bytes).

CSO must be set to the location of TCP or UDP checksum in the packet. *CSS* must be set to the beginning of the IP header or the L4 (TCP/UDP) header. Checksum calculation is not done if *CSO* or *CSS* are out of range. This occurs if (*CSS* > length) OR (*CSO* > length - 1).

For the 802.1Q header, the offset values depend on the VLAN insertion enable bit — the *VLE* bit. If they are not set (VLAN tagging included in the packet buffers), the offset values should include the VLAN tagging. If these bits are set (VLAN tagging is taken from the packet descriptor), the offset values should exclude the VLAN tagging.

Hardware does not add the 802.1q EtherType or the VLAN field following the 802.1Q EtherType to the checksum. So for VLAN packets, software can compute the values to back out only on the encapsulated packet rather than on the added fields.

Note: UDP checksum calculation is not supported by the legacy descriptor because the legacy descriptor does not support the translation of a checksum result of 0x0000 to 0xFFFF needed to differentiate between an UDP packet with a checksum of zero and an UDP packet without checksum.

Because the CSO field is eight bits wide, it puts a limit on the location of the checksum to 255 bytes from the beginning of the packet.

Note: CSO must be larger than CSS.

Software must compute an offsetting entry to back out the bytes of the header that should not be included in the TCP checksum and store it in the position where the hardware computed checksum is to be inserted.

Hardware adds the checksum at the byte offset indicated by the CSO field. Checksum calculations are for the entire packet starting at the byte indicated by the CSS field. The byte offset is counted from the first byte of the packet fetched from host memory.

Command Byte — CMD (8)

The CMD byte stores the applicable command and has the fields listed in [Table 7-35](#).

Table 7-35. Transmit Command (TDESC.CMD) Layout

7	6	5	4	3	2	1	0
RSV	VLE	DEXT	RSV	RS	IC	IFCS	EOP

RSV (bit 7)

Reserved.

VLE (bit 6) - VLAN Packet Enable

When set to 1b, VLE indicates that the packet is a VLAN packet and hardware adds the VLAN header to the Tx packet. The VLAN EtherType is taken from DMATXCTL.VT and the 802.1q VLAN tag is taken from the VLAN field in the Tx descriptor. See [Section 7.4.5](#) for details about double VLAN.

Table 7-36. VLAN Tag Insertion Decision Table for VLAN Mode Enabled

VLE	Action
0	Send generic Ethernet packet.
1	Send 802.1Q packet; the <i>Ethernet Type</i> field comes from the VET field of the VLNCTRL register and the VLAN data comes from the VLAN field of the Tx descriptor.

Note: This table is relevant only if PFVMVIR.VLANA = 00b (use descriptor command) for the queue.

DEXT (bit 5) - Descriptor Extension

Zero for legacy mode.

RSV (bit 4)

Reserved.

RS (bit 3) - Report Status

RS signals hardware to report the DMA completion status indication as well as triggering ITR. Hardware indicates a DMA completion by setting the *DD* bit in the transmit descriptor when *TDWBAL[n].HEAD_WB_EN* = 0b or by Head Write-back if *HEAD_WB_EN* = 1b (see [Section 7.2.3.5.2](#)). The *RS* bit is permitted only on descriptors that has the *EOP* bit set (last descriptor of a packet).

Note: Software should not set the *RS* bit when *TXDCTL.WTHRESH* is greater than zero. Instead, the hardware reports the DMA completion according to the *WTHRESH* rules (explained in [Section 7.2.3.5.1](#)). This note is relevant only for descriptor write back while in head write-back mode. *WTHRESH* must also be set to zero.

When *TXDCTL.WTHRESH* = zero, software must set the *RS* bit on the last descriptor of every packet.

There are some exceptions for descriptor completion indication in head write-back mode. For more details see [Section 7.2.3.5.2](#).

IC (bit 2) - Insert Checksum

Hardware inserts a checksum at the offset indicated by the *CSO* field if the *Insert Checksum* bit (*IC*) is set.

IFCS (bit 1) - Insert FCS

When set, hardware appends the MAC FCS at the end of the packet. When cleared, software should calculate the FCS for proper CRC check. There are several cases in which software must set IFCS as follows:

- Transmitting a short packet while padding is enabled by the *HLREG0.TXPADEN* bit.
- Checksum offload is enabled by the *IC* bit in the *TDESC.CMD*.
- VLAN header insertion enabled by the *VLE* bit in the *TDESC.CMD* or by the *PFVMVIR.VLANA* fields.
- E-tag insertion enabled by the *PFVMVIR.TAGA* fields.
- TSO or TCP/IP checksum offload using a context descriptor.

Note: TSO offload are relevant only to advanced Tx descriptors.

EOP (bit 0) - End of Packet

A packet can be composed of multiple buffers (each of them indicated by its own descriptor). When *EOP* is set, it indicates the last descriptor making up the packet.

Note: *VLE*, *IFCS*, and *IC* fields should be set in the first descriptor of a packet. The *RS* bit can be set only on the last descriptor of a packet. The *DEXT* bit must be set to zero for all descriptors. The *EOF* bit is meaningful in all descriptors.

STA (4) — Transmitted Status

DD (bit 0) — Descriptor Done Status

This bit provides a status indication that the DMA of the buffer has completed. Software might re-use descriptors with the *DD* bit set and any other descriptors processed by the hardware before this one. The other bits in the *STA* field are reserved.

Rsvd — Reserved (4)

VLAN (16)

The *VLAN* field is used to provide the 802.1q/802.1ac tagging information. The *VLAN* field is qualified on the first descriptor of each packet when the *VLE* bit is set to 1b. The *VLAN* field is provided in network order and is meaningful in the first descriptor of a packet. See [Section 7.2.3.2.1](#) for more details.

Table 7-37. VLAN Field (TDESC.VLAN) Layout

15	13	12	11	0
PRI	DEI	VLAN		

7.2.3.2.3 Advanced Transmit Context Descriptor

	63	56	55	48	47	42	41	32	31	16	15	9	8	0				
0	TUNNELLEN		OUTERIPLEN							VLAN			MACLEN					
8	MSS				L4LEN		RSV	IDX	Reserved must be 0x3F	DEXT	RSV	DTYP	TUCMD					
	63	48	47	40	39	37	36	35	30	29	28	24	23	20	19	9	8	0

IPLEN (9)

This field holds the value of the IP header length for the IP checksum offload feature. If an offload is requested, *IPLEN* must be greater than or equal to 20, and less than or equal to 450. For IP tunnel packets (IPv4-IPv6) *IPLEN* must be defined as the length of the two IP headers. The hardware is able to offload the L4 checksum calculation while software should provide the IPv4 checksum.

MACLEN (7)

This field indicates the length of the MAC header. When an offload is requested, one of the *TSE* bits (in the advanced transmit data descriptor) or *IXSM* bit or *TXSM* bit are set, *MACLEN* must be larger than or equal to 14, and less than or equal to 127. This field should include only the part of the L2 header supplied by the driver and not the parts added by hardware. The following table lists the value of *MACLEN* in the different cases.

Regular VLAN	Extended tag	MACLEN
By hardware or none	None	14
By hardware or none	VLAN	18
By hardware or none	E-tag	22
By software	None	18
By software	VLAN	22
By software	E-tag	26

VLAN (16)

This field contains the 802.1Q VLAN tag to be inserted in the packet during transmission. This VLAN tag is inserted when a packet using this context has its *DCMD.VLE* bit is set. This field should include the entire 16-bit *VLAN* field including DEI and priority fields as listed in [Table 7-37](#).

Note: The *VLAN* field is provided in network order. See [Section 7.2.3.2.1](#).

OUTERIPLEN (8)

This field holds the value of the outer IP header length.

TUNNELLEN (8)

This field holds the length value of the tunnel headers including the inner L2 header.

TUCMD (11)

RSV (bit 10-7)

Reserved.

Reserved (bit 6)

Reserved (bit 5:4)

Reserved.

L4T (bit 3:2)

L4 Packet TYPE (00: UDP; 01: TCP; 10: SCTP; 11: RSV).

IPV4(bit 1)

IP Packet Type: When 1b, IPv4; when 0b, IPv6.

DTYP (4)

This field is always 0010b for this type of descriptor.

RSV(1)

Reserved.

DEXT (1)

Descriptor Extension (one for advanced mode).

Reserved (6)

Reserved.

IDX (1)

The context descriptor is posted to a context table in hardware. There are two context tables per queue. The *IDX* is the index of the context tables.

Note: Because the E610 supports only two context descriptors per queue, the two MS bits are reserved and should be set to 0b.

RSV(1)

Reserved.

L4LEN(8)

This field holds the layer 4 header length. If *TSE* is set, this field must be greater than or equal to 8 and less than or equal to 64. Otherwise, this field is ignored. Note that for UDP segmentation the L4 header size equals 8 and for TCP segmentation (with no TCP options) it equals 20.

MSS (16)

This field controls the Maximum Segment Size. This specifies the maximum protocol payload segment sent per frame, not including any header. MSS is ignored when *DCMD.TSE* is not set.

7.2.3.2.3.1 TCP/UDP Segmentation

The total length of each frame (or segment) excluding Ethernet CRC as follows. Note that the last packet of a TCP segmentation might be shorter.

$$\text{MACLEN} + 4(\text{if VLE set}) + \text{OUTERIPLEN} + \text{TUNNELLEN} + \text{IPLEN} + \text{L4LEN} + \text{MSS}$$

Note: The headers lengths must meet the following: $\text{MACLEN} + \text{IPLEN} + \text{L4LEN} \leq 512$

The context descriptor requires valid data only in the fields used by the specific offload options. Table 7-38 lists the required valid fields according to the different offload options.

Table 7-38. Valid Fields by Offload Option

	Context Fields	VLAN	MACLEN	OUTERIPLEN/TUNNELLEN	IPLEN	L4LEN		IPV4	L4T	Encryption	MSS	QCNTLEN	CC (data descriptor)
Required Offload	VLAN insertion	yes										yes	
	IPv4 XSUM		yes		yes			1				yes	0
	L4 XSUM		yes		yes				yes			yes	0
	TCP/UDP Seg		yes		yes	yes		yes	yes		yes	yes	0
	Tunnel XSUM		yes	yes	yes				yes				1
	Tunnel Seg		yes	yes	yes	yes		yes	yes		yes	yes	1
	Tx switch		yes		yes	yes		yes	yes	n/a		yes	1

Note: All fields that are not used in the context descriptor must be set to zero.

7.2.3.2.4 Advanced Transmit Data Descriptor

Table 7-39. Advanced Transmit Data Descriptor Read Format

	63	46	45	40	39	38 36	35 32	31 24	23 20	19 18	17 16	15	0
0	Address[63:0]												
8	PAYLEN		POPTS		CC	IDX	STA	DCMD	DTYP	MAC	TUNNEL	DTALEN	

Table 7-40. Advanced Transmit Data Descriptor Write-back Format

	63	36	35	32	31	0
0	RSV					
8	RSV			STA		RSV

Address (64)

This field holds the physical address of a data buffer in host memory, which contains a portion of a transmit packet. This field is meaningful in all descriptors.

DTALEN (16)

This field holds the length in bytes of data buffer at the address pointed to by this specific descriptor. This field is meaningful in all descriptors. The maximum length is 16 KB with no limitations on the minimum size. Refer to the comment on descriptors with zero length described in the sections that follow.

TUNNEL (2)

Tunnel Type (bit 0):

- 0 = VXLAN
- 1 = NVGRE

OUTERIPCS (bit 1) - Outer IP checksum enable

Indicates this is a tunnel packet and that the *OUTERIPLEN/TUNNELLEN* parameters in the context descriptor are valid.

MAC (2)

This field is meaningful on the first descriptor of the packet(s).

Reserved (bit 0)

Reserved.

1588 (bit 1)

IEEE1588 time stamp packet.

DTYP (4)

0011b for advanced data descriptor. DTYP should be valid in all descriptors of the packet(s).

DCMD (8)

TSE (bit 7) - Transmit Segmentation Enable

This bit indicates a TCP segmentation request. When *TSE* is set in the first descriptor of a TCP packet, hardware must use the corresponding context descriptor to perform segmentation.

Note: It is recommended that `HLREG0.TXPADEN` be enabled when TSE is used since the last frame can be shorter than 60 bytes — resulting in a bad frame.

VLE (bit 6) - VLAN Packet Enable

This bit indicates that the packet is a VLAN packet (hardware must add the VLAN EtherType and an 802.1q VLAN tag to the packet).

DEXT (bit 5) - Descriptor Extension

This bit must be one to indicate advanced descriptor format (as opposed to legacy).

Rsv (bit 4)

Reserved.

RS (bit 3) - Report Status

See the description in the legacy transmit descriptor in [Section 7.2.3.2.2](#).

Rsv (bit 2)

Reserved.

IFCS (bit 1) - Insert FCS

When this bit is set, the hardware appends the MAC FCS at the end of the packet. When cleared, software should calculate the FCS for proper CRC check. There are several cases in which software must set IFCS as follows:

- Transmitting a short packet while padding is enabled by the `HLREG0.TXPADEN` bit.
- Checksum offload is enabled by the either `IC`, `TXSM` or `IXSM` bits in the `TDESC.DCMD`.
- VLAN header insertion enabled by the `VLE` bit in the `TDESC.DCMD`.
- TCP segmentation offload enabled by the `TSE` bit in the `TDESC.DCMD`.

EOP (bit 0) - End of Packet

A packet might be composed of multiple buffers (each of them is indicated by its own descriptor). When `EOP` is set, it indicates the last descriptor making up the packet. In transmit segmentation (explained later on in this section) the `EOP` flag indicates the last descriptor of the last packet of the segmented transmission.

Note: `TSE`, `VLE`, and `IFCS` fields should be set in the first descriptor of the packet(s). The `RS` bit can be set only on the last descriptor of the packet. The `EOP` bit is valid in all descriptors. The `DEXT` bit must be set to 1b for all descriptors.

Descriptors with zero length, transfer no data. If the `RS` bit in the command byte is set, the `DD` field in the status word is not written when hardware processes them.

STA (4)

Rsv (bit 3:1)

Reserved.

DD (bit 0) - Descriptor Done

The `DD` bit provides a status indication that the DMA of the buffer has completed. Software might re-use descriptors with the `DD` bit set, and any other descriptors processed by hardware before this one. In TSO, the buffers that include the TSO header are used multiple times during transmission and special considerations should be made as described in [Section 7.2.4.2.2](#).

IDX (3)

This field holds the index into the hardware context table to indicate which of the two per-queue contexts should be used for this request. If no offload is required and the *CC* bit is cleared, this field is not relevant and no context needs to be initiated before the packet is sent. See [Table 7-38](#) for details of which packets requires a context reference. This field is relevant only on the first descriptor of the packet(s).

CC (1)

Check Context bit — When set, a Tx context descriptor indicated by *IDX* index should be used for this packet(s). The *CC* bit should be set in the following cases:

1. Non-zero *QCNTLEN* field is required (defined in the context descriptor).
2. Tx switching is enabled (including anti-spoof checks).

POPTS (6)

Rsv (bit 5:2)

Reserved.

TXSM (bit 1) - Insert TCP/UDP Checksum

When set to 1b, the L4 checksum must be inserted. In this case, *TUCMD.LP4* indicates whether the checksum is TCP or UDP or SCTP. When *DCMD.TSE* is set, *TXSM* must be set to 1b. If this bit is set, the packet should at least contain a TCP header.

IXSM (bit 0) - Insert IP Checksum

This field indicates that IP checksum must be inserted. In IPv6 mode, it must be reset to 0b. If *DCMD.TSE* and *TUCMD.IPV4* are set, *IXSM* must be set to 1b. If this bit is set, the packet should at least contain an IP header.

This field is relevant only on the first descriptor of the packet(s).

PAYLEN (18)

PAYLEN indicates the size (in byte units) of the data buffer(s) in host memory for transmission.

In a single-send packet, *PAYLEN* defines the entire packet size fetched from host memory. It does not include the fields that hardware adds such as: optional VLAN tagging and Ethernet CRC or Ethernet padding.

In TSO (regardless if it is transmitted on a single or multiple packets), the *PAYLEN* defines the protocol payload size fetched from host memory.

- In TCP or UDP segmentation offload, *PAYLEN* defines the TCP/UDP payload size.

This field is relevant only on the first descriptor of the packet(s). The minimum transmitted packet size excluding VLAN padding and CRC bytes is 17 and the *PAYLEN* size should meet this limitation. On a single-packet send, the maximum size of the *PAYLEN* is dictated by the maximum allowed packet size which is 15.5 KB. On TSO, the maximum *PAYLEN* can be up to $2^{18}-1$.

Note: When a packet spreads over multiple descriptors, all of the descriptor fields are valid only on the first descriptor of the packet, except for *RS* and *EOP* bits, which are set on the last descriptor of the packet.

7.2.3.3 Transmit Descriptor Ring

The transmit descriptor ring structure (shown in Figure 7-20) uses a contiguous memory space. A set of four registers (described later in this section) maintain the transmit descriptor ring in the host memory. Hardware maintains internal circular queues of 64 descriptors per queue to hold the descriptors that were fetched from the software ring.

Descriptors handled to hardware should not be manipulated by software until hardware completes its processing. It is indicated by advancing the head pointer beyond these descriptors.

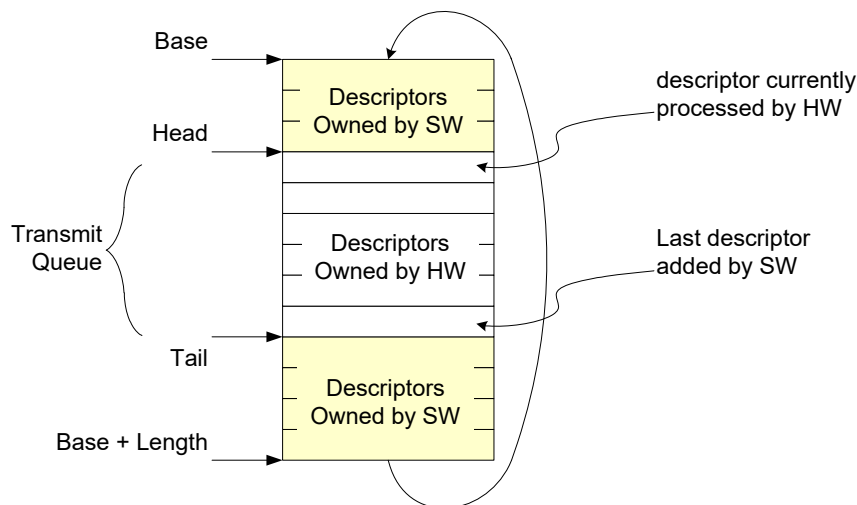


Figure 7-20. Transmit Descriptor Ring Structure

The transmit descriptor ring is defined by the following registers:

- **Transmit Descriptor Base Address register (TDBA 0-127)** — This register indicates the start address of the descriptor ring buffer in the host memory; this 64-bit address is aligned on a 128-byte boundary and is stored in two consecutive 32-bit registers. Hardware ignores the lower 7 bits.
- **Transmit Descriptor Length register (TDLEN 0-127)** — This register determines the number of bytes allocated to the circular buffer. This value must be 0 modulo 128.
- **Transmit Descriptor Head register (TDH 0-127)** — This register holds a value that is an offset from the base and indicates the in-progress descriptor. There can be up to 64 K minus 8 descriptors in the circular buffer. The transmit queue consists of the descriptors between the head and tail pointers. Transmission starts with the descriptor pointer by the head registers. When the DMA engine processes a descriptor, it might optionally write back the completed descriptor and then advance the head pointer. It then processes the next descriptor up to the point that the head pointer reaches the tail. Head equals tail means that the transmit queue in host memory is empty. Reading this register indicates the hardware progress to the software. All descriptors behind the head pointer and in front of tail register are owned by the software. The other descriptors are owned by the hardware and should not be modified by the software.

- **Transmit Descriptor Tail register (TDT 0-127)** — This register holds a value, which is an offset from the base, and indicates the location beyond the last descriptor hardware can process. Software adds new descriptors to the ring by writing descriptors in the circular buffer pointed by the tail pointer. The new descriptor(s) are indicated to hardware by updating the tail pointer one descriptor above the last added descriptor. Note that a single packet or TSO might be composed of multiple descriptors. The transmit tail pointer should never point to the middle of a packet or TSO, which might cause undesired software/hardware races.

Software might detect which packets have already been processed by hardware using the following:

- Read the TDH head register to determine which packets (those logically before the head) have been transferred to the on-chip FIFO or transmitted. This method is not recommended as races between the internal update of the head register and the actual write back of descriptors can occur.
- When head write back is enabled ($TDWBAL[n].HEAD\_WB\_EN = 1b$) software might read the image of the head pointer in host memory at the address defined by $TDWBAH[n]/TDWBAL[n]$ pair. Hardware updates the head image in host memory by completed descriptors as described in [Section 7.2.3.5.2](#).
- When head write back is not enabled ($TDWBAL[n].HEAD\_WB\_EN = 0b$), software might track the *DD* bits in the descriptor ring. Descriptor write back is controlled by the *RS* bit and the *WTHRESH* setting as well as interrupt assertion.
- Issue an interrupt. An interrupt condition is generated each time a packet was transmitted or received and a descriptor was write back or transmit queue goes empty ($EICR.RTxQ[0-19]$). This interrupt can either be enabled or masked.

All of the registers controlling the descriptor rings behavior should be set before transmit is enabled.

7.2.3.4 Transmit Descriptor Fetching

The E610 fetches new descriptors as required for packet transmission depending on its on-die descriptor buffer state:

- **Fetch** — The on-chip descriptor buffer is empty or contains less descriptors than a complete packet.
 - A fetch starts as soon as any descriptors are made available (host writes to the tail pointer).
 - A request is issued for any available descriptors up to the size of the on-die buffer.
 - Once the sum of on-die descriptors and requested descriptors is more than required for a single packet, the buffer transitions to the pre-fetch state.
 - If several on-chip descriptor queues are empty simultaneously, queues are served in round robin arbitration.
- **Pre-Fetch** — The on-chip descriptor buffer becomes almost empty while there are enough descriptors in the host memory.
 - The on-chip descriptor buffer is defined as almost empty if it contains less descriptors than the threshold defined by $TXDCTL[n].PTHRESH$.
 - The transmit descriptor contains enough descriptors if it includes more ready descriptors than the threshold defined by $TXDCTL[n].HTHRESH$.
 - In pre-fetch mode, descriptors are fetched only after there are no other DMA activity of greater priority as: transmit descriptor fetch; status write-backs or packet data transfers).
 - A request is issued for any available descriptors up to the capacity of the on-die buffer.

- If several on-chip descriptor queues are in this situation simultaneously, queues are served in round robin arbitration.

- **Idle** — Requests are not issued. This is the state reached when none of the previous states apply.

Note: Software must update the Tail register on packet boundaries. That is, the last valid descriptor might not be a context descriptor and must have the *EOP* bit set.

7.2.3.4.1 Transmit Descriptor Fetch and Write-back Settings

This section describes the settings of transmit descriptor thresholds. It relates to fetch thresholds described above as well as the write-back threshold (*WTHRESH*) when operating in descriptor write-back mode, which is described in [Section 7.2.3.5.1](#).

- Transmit descriptor fetch setting is programmed in the *TXDCTL[n]* register per queue. The default settings of *PTHRESH*, *HTHRESH* and *WTHRESH* are zero's.
- To reduce transmission latency, it is recommended to set the *PTHRESH* value as high as possible while the *HTHRESH* and *WTHRESH* as low as possible (down to zero).
- To minimize PCIe overhead the *PTHRESH* should be set as low as possible while *HTHRESH* and *WTHRESH* should be set as high as possible.
- The sum of *PTHRESH* plus *WTHRESH* must not be greater than the on-chip descriptor buffer size.
- Some practical rules:
 - CPU cache line optimization — Assume 'N' equals the CPU cache line divided by 16 (descriptor size). Then, to align descriptors pre-fetch to CPU cache line (in most cases), it is advised to set *PTHRESH* to the on-chip descriptor buffer size minus 'N' and *HTHRESH* to 'N'. To align descriptor write back to the CPU cache line it is advised to set *WTHRESH* to either 'N' or even 2 times 'N'. Note that partial cache line writes might significantly degrade performance. Therefore, it is highly recommended to follow this advice.
 - Minimizing PCIe overhead — As an example, setting *PTHRESH* to the on-chip descriptor buffer size minus 16 and *HTHRESH* to 16 minimizes the PCIe request and header overhead to ~20% of the bandwidth required for the descriptor fetch.
 - Minimizing transmission latency from tail update — Setting *PTHRESH* to the on-chip descriptor buffer size minus 'N' ('N' previously defined) while *HTHRESH* and *WTHRESH* to zero.
 - Threshold settings in DCB mode — Note that only values of *PTHRESH* equals on-chip descriptor buffer size minus 8 and *HTHRESH* equals 4 were thoroughly tested.

Note: As previously described, device setting is a trade-off between overhead (translated to performance) and latencies. It is expected that some level of optimization is done at software driver development phase. Customers who want better performance might need to adjust the threshold values according to the previous guidelines while optimizing to specific platform and targets.

7.2.3.5 Transmit Write Back

The E610 periodically updates software on its progress in processing transmit buffers. Two methods are described for doing so:

- Updating by writing back into the Tx descriptor.
- Update by writing to the head pointer in system memory.

7.2.3.5.1 Tx Descriptor Write Back

When the `TXDCTL[n].WTHRESH` equals zero, descriptors are written back for those descriptors with the *RS* bit set. When the `TXDCTL[n].WTHRESH` value is greater than zero, descriptors are accumulated until the number of accumulated descriptors equals the `TXDCTL[n].WTHRESH` value, then these descriptors are written back. Accumulated descriptor write back enables better use of the PCIe bus and memory bandwidth.

Any descriptor write back includes the full 16 bytes of the descriptor.

Descriptors are written back in one of three cases:

- `TXDCTL[n].WTHRESH = 0` and a descriptor that has *RS* set is ready to be written back.
- `TXDCTL[n].WTHRESH > 0` and `TXDCTL[n].WTHRESH` descriptors have accumulated.
- `TXDCTL[n].WTHRESH > 0` and the corresponding *EITR* counter has reached zero. The timer expiration flushes any accumulated descriptors and sets an interrupt event (*TXDW*).

An additional mode in which transmit descriptors are not written back at all and the head pointer of the descriptor ring is written instead is described in the following section.

7.2.3.5.2 Tx Head Pointer Write Back

In legacy hardware, transmit requests are completed by writing the *DD* bit to the transmit descriptor ring. This causes cache trash since both the driver and hardware are writing to the descriptor ring in host memory. Instead of writing the *DD* bits to signal that a transmit request is complete, hardware can write the contents of the descriptor queue head to host memory. The driver reads that memory location to determine which transmit requests are complete. To improve the performance of this feature, the driver needs to program *TPH* registers to configure which CPU processes each Tx queue.

The head pointer is reflected in a memory location that is allocated by software for each queue.

Rules for head pointer write back:

- Head write back occurs if `TDWBAL[n].HEAD_WB_EN` is set for this queue, and the *RS* bit is set in the Tx descriptor, following its corresponding data upload into packet buffer.
 - If the head write-back feature is enabled, software must set *WTHRESH* to 0x0 while only descriptors with the *RS* bit set, generate header write back.
 - Note that the head pointer write back does not hold transmission. Instead, if packets with the *RS* bit are transmitted fast enough, it might happen that the header pointer write back is not updated for each and every packet. In addition, it might happen that the head pointer write back might be updated up to descriptors that do not have the *RS* bit set. In such cases, hardware might report a completion of a descriptor that might not be the last descriptor in a TSO or even the last descriptor in a single packet.

The driver has control of this feature per queue through the *TDWBAL* and *TDWBAH* registers.

The low register's LSB hold the control bits.

- The *HEAD_WB_EN* bit enables activation of tail write back. In this case, no descriptor write back is executed.
- The 30 upper bits of this register hold the lowest 32 bits of the head write-back address, assuming that the two last bits are zero.

The high register holds the high part of the 64-bit address.

Note: Hardware writes a full DWord when writing this value, so software should reserve enough space for each head value and make sure the *TDBAL* value is DWord-aligned.

7.2.4 TCP and UDP Segmentation

Hardware TCP segmentation is one of the offloading options supported by the Windows* and Linux* TCP/IP stack. This is often referred to as Large Send offloading or TSO. This feature enables the TCP/IP stack to pass to the network device driver a message to be transmitted that is bigger than the Maximum Transmission Unit (MTU) of the medium. It is then the responsibility of the device driver and hardware to divide the TCP message into MTU size frames that have appropriate layer 2 (Ethernet), 3 (IP), and 4 (TCP) headers. These headers must include sequence number, checksum fields, options and flag values as required. Note that some of these values (such as the checksum values) are unique for each packet of the TCP message, and other fields such as the source IP Address is constant for all packets associated with the TCP message.

Similar to TCP segmentation, the E610 also provides a capability to offload UDP segmentation. Note that current UDP segmentation offload is not supported by any standard operating system.

Note: CRC appending (HLREG0.TXCRCEN) must be enabled in TCP / UDP segmentation mode because CRC is inserted by hardware.

Padding (HLREG0.TXPADEN) must be enabled in TCP / UDP segmentation mode, since the last frame might be shorter than 60 bytes — resulting in a bad frame if *TXPADEN* is disabled.

The offloading of these mechanisms to the device driver and the E610 saves significant CPU cycles. The device driver shares the additional tasks to support these options with the E610.

7.2.4.1 Assumptions and Restrictions

The following assumptions apply to the TCP / UDP segmentation implementation in the E610:

- To limit the internal cache dimensions, software is required to spread the header onto a maximum four descriptors, while still allowed to mix header and data in the last header buffer. This limitation stands for up to Layer 4 header included, and for IPv4 or IPv6 independently.
- The maximum size of a single TSO can be as large as defined by the *PAYLEN* field in the Tx data descriptor (such as up to 256 KB).
- The *RS* bit operation is not changed. Interrupts are set after data in the buffers pointed to by individual descriptors is transferred (DMA'ed) to hardware.
- SNAP packets are not supported for segmentation.
- IP in IP tunneled packets are not supported for offloading under TSO operation. VXLAN and NVGRE tunneled packets can be segmented.
- Software must enable the Ethernet CRC offload in the HLREG0.TXCRCEN register since CRC must be inserted by hardware after the checksum has been calculated.
- Software must initialize the appropriate checksum fields in the packet's header.

7.2.4.2 Transmission Process

The transmission process involves the following:

- The protocol stack receives from an application a block of data that is to be transmitted.
- The protocol stack calculates the number of packets required to transmit this block based on the MTU size of the media and required packet headers.

- The stack interfaces with the device driver and passes the block down with the appropriate header information: Ethernet, IP and TCP / UDP headers.
- The stack interfaces with the device driver and commands the driver to send the individual packet. The device driver sets up the interface to the hardware (via descriptors) for the TCP / UDP segmentation.
- The hardware transfers (DMA's) the packet data and performs the Ethernet packet segmentation and transmission based on offset and payload length parameters in the TCP/IP or UDP/IP context descriptor including:
 - Packet encapsulation
 - Header generation and field updates including IPv4/IPv6 and TCP/UDP checksum generation.
- The driver returns ownership of the block of data to the NOS when the hardware has completed the DMA transfer of the entire data block.

7.2.4.2.1 TCP and UDP Segmentation Data Fetch Control

To perform TCP / UDP segmentation in the E610, the DMA must be able to fit at least one packet of the segmented payload into available space in the on-chip packet buffer. The DMA does various comparisons between the remaining payload and the packet buffer available space, fetching additional payload and sending additional packets as space permits.

The E610 enables interleaving between different TSO requests at an Ethernet packet level. In other words, the E610 might fetch part of a TSO from a queue, equivalent to one or more Ethernet packets, then transition to another queue and fetch the equivalent of one or more packets (TSO or not), then move to another queue (or the first queue), etc. The E610 decides on the order of data fetched based on its QoS requirements (such as bandwidth allocation and priority).

To enable interleaving between descriptor queues at the Ethernet frame resolution inside TSO requests, the frame header pointed by the so called header descriptors are re-read from system memory for every TSO segment (once per packet), storing in an internal cache only the header's descriptors instead of the header's content.

7.2.4.2.2 TCP and UDP Segmentation Write-back Modes

TCP / UDP segmentation mode uses the buffers that contain the header of the packet multiple times (once for each transmitted segment). Software should guarantee that the header buffers are available throughout the entire TSO transmission. Therefore, software should not re-use any descriptors of the TSO header during the TSO transmission.

7.2.4.3 TCP and UDP Segmentation Performance

Performance improvements for a hardware implementation of TCP / UDP segmentation offload include:

- The stack does not need to partition the block to fit the MTU size, saving CPU cycles.
- The stack only computes one Ethernet, IP, and TCP / UDP header per segment, saving CPU cycles.
- The stack interfaces with the device driver only once per block transfer, instead of once per frame.
- Larger PCI bursts are used, which improves bus efficiency (such as lowering transaction overhead).
- Interrupts are easily reduced to one per TCP / UDP message instead of one per packet.
- Fewer I/O accesses are required to command the hardware.

7.2.4.4 Packet Format

A TCP / UDP message can be as large as 256 KB and is generally fragmented across multiple pages in host memory. The E610 partitions the data packet into standard Ethernet frames prior to transmission. The E610 supports calculating the Ethernet, IP, TCP, and UDP headers, including checksum, on a frame-by-frame basis. For tunneled packets (NVGRE, VXLAN), The E610 also supports updating the outer IPv4 header.

Table 7-41. TCP/IP and UDP/IP Packet Format Sent by Host

Pseudo Header				Data
Ethernet	Optional tunnel header	IPv4/IPv6	TCP/UDP	DATA (full TCP/UDP message)

Table 7-42. Packets Format Sent by Device

Pseudo Header (updated)	Data (first MSS)	FCS	...	Pseudo Header (updated)	Data (Next MSS)	FCS	...

Frame formats supported by the E610 include:

- Ethernet 802.3
- IEEE 802.1Q VLAN (Ethernet 802.3ac)
- Ethernet Type 2
- NVGRE (see frame format in [Section A.2.5.1](#))
- VXLAN (see frame format in [Section A.2.5.2](#))
- IPv4 headers with options
- IPv6 headers with extensions
- TCP with options
- UDP with options

VLAN tag insertion can be handled by hardware.

Note: UDP (unlike TCP) is not a reliable protocol and fragmentation is not supported at the UDP level. UDP messages that are larger than the MTU size of the given network medium are normally fragmented at the IP layer. This is different from TCP, where large TCP messages can be fragmented at either the IP or TCP layers depending on the software implementation.

The E610 has the ability to segment UDP traffic (in addition to TCP traffic); however, because UDP packets are generally fragmented at the IP layer, the E610's segmentation capability might not be used in practice for UDP.

7.2.4.5 TCP and UDP Segmentation Indication

Software indicates a TCP/UDP segmentation transmission context to the hardware by setting up a TCP/IP or UDP/IP context transmit descriptor (see [Section 7.2.3](#)). The purpose of this descriptor is to provide information to the hardware to be used during the TCP / UDP segmentation offload process.

Setting the *TSE* bit in the *DCMD* field to one (in the data descriptor) indicates that this descriptor refers to the segmentation context (as opposed to the normal checksum offloading context). This causes the checksum offloading, packet length, header length, and maximum segment size parameters to be loaded from the descriptor into the device.

The TCP/UDP segmentation prototype header is taken from the packet data itself. Software must identify the type of packet that is being sent (IPv4/IPv6, TCP/UDP, other), calculate appropriate checksum off loading values for the desired checksums, and then calculate the length of the header that is pre-appended. The header can be up to 512 bytes in length.

Once the TCP/UDP segmentation context has been set, the next descriptor provides the initial data to transfer. This first descriptor(s) must point to a packet of the type indicated. Furthermore, the data it points to might need to be modified by software as it serves as the prototype header for all packets within the TC /UDP segmentation context. The following sections describe the supported packet types and the various updates that are performed by hardware. This should be used as a guide to determine what must be modified in the original packet header to make it a suitable prototype header.

The following summarizes the fields considered by the driver for modification in constructing the prototype header.

IP Header

For IPv4 headers:

- Identification field should be set as appropriate for first packet of send (if not already).
- Header checksums of inner and outer IP headers should be zeroed out unless some adjustment is needed by the driver.

TCP Header

- Sequence number should be set as appropriate for first packet of send (if not already).
- PSH, and FIN flags should be set as appropriate for LAST packet of send.
- TCP checksum should be set to the partial pseudo-header checksum as follows (there is a more detailed discussion of this in [Section 7.2.4.6](#)):

Table 7-43. TCP Partial Pseudo-header Checksum for IPv4

IP Source Address		
IP Destination Address		
Zero	Layer 4 Protocol ID	Zero

Table 7-44. TCP Partial Pseudo-header Checksum for IPv6

IPv6 Source Address	
IPv6 Final Destination Address	
Zero	
Zero	Next Header

UDP Header

- Checksum should be set as in TCP header, as previously explained.

The following sections describe the updating process performed by the hardware for each frame sent using the TCP segmentation capability.

7.2.4.6 Transmit Checksum Offloading with TCP and UDP Segmentation

The E610 supports checksum offloading as a component of the TCP/UDP segmentation off-load feature and as stand-alone capability. [Section 7.2.5](#) describes the interface for controlling the checksum off-loading feature. This section describes the feature as it relates to TCP/UDP segmentation.

The E610 supports IP and TCP header options in the checksum computation for packets that are derived from the TCP segmentation feature.

Two specific types of checksum are supported by the hardware in the context of the TCP/UDP segmentation off-load feature:

- IPv4 checksum
- TCP/UDP checksum

Each packet that is sent via the TCP/UDP segmentation off-load feature optionally includes the IPv4 checksum and/or the TCP/UDP checksum.

All checksum calculations use a 16-bit wide one's complement checksum. The checksum word is calculated on the outgoing data.

Refer to [Table 7-45](#) for the list of supported transmit checksums per packet type.

7.2.4.7 IP/TCP/UDP Header Updating

IP/TCP and IP/UDP header is updated for each outgoing frame based on the header prototype that hardware DMA's from the first descriptor(s). The checksum fields and other header information are later updated on a frame-by-frame basis. The updating process is performed concurrently with the packet data fetch.

The following sections define what fields are modified by hardware during the TCP/UDP segmentation process by the E610.

7.2.4.7.1 TCP/IP/UDP Header for the First Frame

The hardware makes the following changes to the headers of the first packet that is derived from each TCP segmentation context.

Tunnel Header

When tunnel offload is enabled (*Tunnel* bit set in the descriptor) the outer IPv4 header is updated as follows:

- $IP\ Total\ Length = OUTERIPLen + TUNNELLEN + IPLen + L4LEN + MSS$
- Calculates the Outer IP Checksum

IPv4 Header

- $IP\ Total\ Length = MSS + L4LEN + IPLen$
- Calculates the IP Checksum

IPv6 Header

- $Payload\ Length = MSS + L4LEN + IPV6\_HDR\_extension^1$

1. $IPV6\_HDR\_extension$ is calculated as $IPLen - 40$ bytes.

TCP Header

- Sequence Number: The value is the sequence number of the first TCP byte in this frame.
- The flag values of the first frame are set by logic AND function between the flag word in the pseudo header and the DTXTCPFLGL.TCP_FLG_FIRST_SEG. The default values of the DTXTCPFLGL.TCP_FLG_FIRST_SEG are set. The flags in a TSO that ends up as a single segment are taken from the in the pseudo header in the Tx data buffers as is.
- Calculates the TCP checksum.

UDP Header

- Calculates the UDP checksum.

7.2.4.7.2 TCP/IP Header for the Subsequent Frames

The hardware makes the following changes to the headers for subsequent packets that are derived as part of a TCP segmentation context:

$$\text{Number of bytes left for transmission} = \text{PAYLEN} - (\text{N} * \text{MSS})$$

Where N is the number of frames that have been transmitted.

Tunnel Header

When tunnel offload is enabled (*Tunnel* bit set in the descriptor) the outer IPv4 header is updated as follows:

- IP Identification: increased from last value (wrap around based on 16-bit width)
- IP Total Length = OUTERIPLen + TUNNELLEN + IPLen + L4LEN + MSS
- Calculates the Outer IP Checksum

IPv4 Header

- IP Identification: increased from last value (wrap around)
- IP Total Length = MSS + L4LEN + IPLen
- Calculate the IP Checksum

IPv6 Header

- Payload Length = MSS + L4LEN + IPV6_HDR_extension¹

TCP Header

- Sequence Number update: Add previous TCP payload size to the previous sequence number value. This is equivalent to adding the MSS to the previous sequence number.
- The flag values of the subsequent frames are set by logic AND function between the flag word in the pseudo header with the DTXTCPFLGL.TCP_FLG_MID_SEG. The default values of the DTXTCPFLGL.TCP_FLG_MID_SEG are set.
- Calculate the TCP checksum

UDP Header

- Calculates the UDP checksum.

7.2.4.7.3 TCP/IP Header for the Last Frame

Hardware makes the following changes to the headers for the last frame of a TCP segmentation context:

$$\text{Last frame payload bytes} = \text{PAYLEN} - (N * \text{MSS})$$

Tunnel Header

When tunnel offload is enabled (*Tunnel* bit set in the descriptor) the outer IPv4 header is updated as follows:

- IP Identification: increased from last value (wrap around based on 16-bit width)
- IP Total Length = OUTERIPLen + TUNNELLEN + IPLen + L4LEN + last frame payload bytes.
- Calculates the Outer IP Checksum

IPv4 Header

- IP Total length = last frame payload bytes + L4LEN + IPLen
- IP identification: increased from last value (wrap around based on 16-bit width)
- Calculate the IP checksum

IPv6 Header

- Payload length = last frame payload bytes + L4LEN + IPV6_HDR_extension¹

TCP Header

- Sequence number update: Add previous TCP payload size to the previous sequence number value. This is equivalent to adding the MSS to the previous sequence number.
- The flag values of the last frames are set by logic AND function between the flag word in the pseudo header and the DTXTCPFLGH.TCP_FLG_LST_SEG. The default values of the DTXTCPFLGH.TCP_FLG_LST_SEG are set. The flags in a TSO that ends up as a single segment are taken from the in the pseudo header in the Tx data buffers as is.
- Calculate the TCP checksum

UDP Header

- Calculates the UDP checksum.

1. IPV6_HDR_extension is calculated as IPLen — 40 bytes.

7.2.5 Transmit Checksum Offloading in Non-Segmentation Mode

The previous section on TCP/UDP segmentation offload describes the IP/TCP/UDP checksum offloading mechanism used in conjunction with segmentation. The same underlying mechanism can also be applied as a stand-alone checksum offloading. The main difference in a single packet send is that only the checksum fields in the IP/TCP/UDP headers are calculated and updated by hardware.

Before taking advantage of the E610's enhanced checksum offload capability, a checksum context must be initialized. For a single packet send, *DCMD.TSE* should be set to zero (in the data descriptor). For additional details on contexts, refer to [Section 7.2.3.3](#).

Enabling checksum offload, software must also enable Ethernet CRC offload by the *HLREG0.TXCRCEN* since CRC must be inserted by hardware after the checksum has been calculated.

Each checksum operates independently. Insertion of the IP and TCP/UDP checksum for each packet are enabled through the transmit data descriptor *POPTS.TXSM* and *POPTS.IXSM* fields, respectively.

7.2.5.1 IP Checksum

Three fields in the transmit context descriptor set the context of the IP checksum offloading feature:

- *TUCMD.IPV4*
- *IPLLEN*
- *MACLEN*

TUCMD.IPV4=1 specifies that the packet type for this context is IPv4, and that the IP header checksum should be inserted. *TUCMD.IPV4*=0 indicates that the packet type is IPv6 (or some other protocol) and that the IP header checksum should not be inserted.

MACLEN specifies the byte offset from the start of the DMA'ed data to the first byte to be included in the checksum, the start of the IP header. The minimal allowed value for this field is 14. Note that the maximum value for this field is 127. This is adequate for typical applications.

Note: The *MACLEN*+*IPLLEN* value must be less than the total DMA length for a packet. If this is not the case, the results are unpredictable.

IPLLEN specifies the IP header length. Maximum allowed value for this field is 511 bytes.

MACLEN+*IPLLEN* specify where the IP checksum should stop. The sum of *MACLEN*+*IPLLEN* must be smaller equals to the first 638 (127+511) bytes of the packet and obviously must be smaller or equal to the total length of a given packet. If this is not the case, the result is unpredictable.

For IP tunnel packets (IPv4-IPv6), *IPLLEN* must be defined as the length of the two IP headers. Hardware is able to offload the L4 checksum calculation while software should provide the IPv4 checksum.

For NVGRE and VXLAN tunneled packet as defined by the *Tunnel* bit in the descriptor, Outer IPv4 checksum is controlled by *OUTERIPCS* bit and is determined by *MACLEN* and *OUTERIPLLEN* similar to the above description. Inner IPv4 checksum is controlled by *TUCMD.IPV4* and offset determination should be adjusted by hardware such that *MACLEN* should be replaced by (*MACLEN*+*OUTIPLLEN*+*TUNNELLEN*)

The 16-bit IPv4 header checksum is placed at the two bytes starting at *MACLEN*+10.

7.2.5.2 TCP and UDP Checksum

Three to five fields in the transmit context descriptor set the context of the TCP/UDP checksum offloading feature:

- *MACLEN*
- *OUTERIPLEN* (optional)
- *TUNNELLEN* (optional)
- *IPLLEN*
- *TUCMD.L4T*

TUCMD.L4T=01b specifies that the packet type is TCP, and that the 16-bit TCP header checksum should be inserted at byte offset $MACLEN + IPLLEN + 16$. *TUCMD.L4T*=00b indicates that the packet is UDP and that the 16-bit checksum should be inserted starting at byte offset $MACLEN + IPLLEN + 6$. If *Tunnel* bit is set, the checksum is inserted at $MACLEN + OUTERIPLEN + TUNNELLEN + IPLLEN + 6$.

$MACLEN + (OUTERIPLEN + TUNNELLEN) + IPLLEN$ specifies the byte offset from the start of the DMA'ed data to the first byte to be included in the checksum, the start of the UDP/TCP header. See *MACLEN* table in [Section 7.2.3.2.3](#) for its relevant values.

Note: The $MACLEN + (OUTERIPLEN + TUNNELLEN) + IPLLEN + L4LEN$ value must be less than the total DMA length for a packet. If this is not the case, the results are unpredictable.

The TCP/UDP checksum always continues to the last byte of the DMA data.

Note: For non-TSO, software still needs to calculate a full checksum for the TCP/UDP pseudo-header. This checksum of the pseudo-header should be placed in the packet data buffer at the appropriate offset for the checksum calculation.

7.2.5.3 SCTP CRC Offloading

For SCTP packets, a CRC32 checksum offload is provided.

Three fields in the transmit context descriptor set the context of the STCP checksum offloading feature:

- *MACLEN*
- *IPLLEN*
- *TUCMD.L4T*

TUCMD.L4T=10b specifies that the packet type is SCTP, and that the 32-bit STCP CRC should be inserted at byte offset $MACLEN + IPLLEN + 8$.

$IPLLEN + MACLEN$ specifies the byte offset from the start of the DMA'ed data to the first byte to be included in the checksum, the start of the STCP header. The minimal allowed value for this sum is 26. See *MACLEN* table in [Section 7.2.3.2.3](#) for its relevant values.

The SCTP CRC calculation always continues to the last byte of the DMA data.

The SCTP total L3 payload size ($PAYLEN - IPLLEN - MACLEN$) should be a multiple of four bytes (SCTP padding not supported).

Note: TSO is not available for SCTP packets.

Note: The CRC field of the SCTP header must be set by driver to zero prior to requesting a CRC calculation offload.

7.2.5.4 Checksum Supported per Packet Types

Table 7-45 lists which checksums are supported per packet type.

Note: TSO is not supported for packet types for which IP checksum and TCP/UDP checksum cannot be calculated.

Table 7-45. Checksums Supported by Packet Type

Packet Type	Hardware IP Checksum Calculation	Hardware TCP/UDP/SCTP Checksum Calculation
IPv4 packets	Yes	Yes
IPv6 packets	No (n/a)	Yes
IPv6 packet with next header options: - Hop-by-hop options - Destinations options - Routing (with len 0) - Routing (with len >0) - Fragment - Home option - Security option (AH/ESP)	No (n/a) No (n/a) No (n/a) No (n/a) No (n/a) No (n/a) No (n/a)	Yes Yes Yes No No No Yes
IPv4 tunnels: - IPv4 packet in an IPv4 tunnel - IPv6 packet in an IPv4 tunnel	No No	No Yes
IPv6 tunnels: - IPv4 packet in an IPv6 tunnel - IPv6 packet in an IPv6 tunnel	No No	No No
Packet is an IPv4 fragment	Yes	No
Packet has 802.3ac tag	Yes	Yes
Packet has TCP or UDP options	Yes	Yes
IP header's protocol field contains protocol # other than TCP, UDP, or SCTP	Yes	No
NVGRE: - Inner IPv4 packet - Inner IPv6 packet	Yes (Inner and outer) Yes (outer)	Yes Yes
VXLAN: - Inner IPv4 packet - Inner IPv6 packet	Yes (Inner and outer) Yes (outer)	Yes (inner only) ¹ Yes (inner only)

1. The outer UDP header of VXLAN packets do not use a checksum.

7.2.6 Transmit Statistics

7.2.6.1 General Notes

- All Statistics registers are cleared on read. In addition, they stick at 0xFF...F when the maximum value is reached.
- Due to divergent paths between interrupt-generation and logging of relevant statistics counts, it might be possible to generate an interrupt to the system for a noteworthy event prior to the associated statistics count actually being increased. This is extremely unlikely due to expected delays associated with the system interrupt-collection and ISR delay, but might be an explanation

for interrupt statistics values that do not quite make sense. Hardware guarantees that any event noteworthy of inclusion in a statistics count is reflected in the appropriate count within 1 μ s; a small time-delay prior to reading the statistics might be required to avoid a potential mismatch between and interrupt and its cause.

- If TSO is enabled, statistics are collected after segmentation.
- All byte (octet) counters composed of 2 registers can be fetched by two consecutive 32-bit accesses while reading the Low 32-bit register first or a single 64-bit access.

7.2.6.2 Transmit Statistics Hierarchy

Figure 7-21 describes the relations between the packet flow and the different statistic counters.

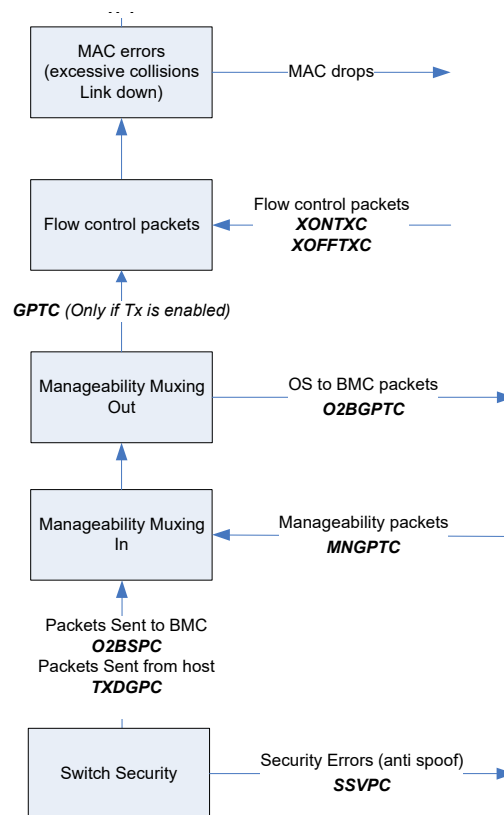


Figure 7-21. Transmit Flow Statistics

7.3 Interrupts

The E610 supports the following interrupt modes. Mapping of interrupts causes is different in each of these modes as described in this section.

- PCI legacy interrupts or MSI or MSI-X and only a single vector is allocated — Selected when `GPIE.MULTIPLE_MSIX` is set to 0b.
- MSI-X with multiple MSI-X vectors in non-IOV mode — Selected when `GPIE.MULTIPLE_MSIX` is set to 1b and `GPIE.VT_MODE` is set to 00b.
- MSI-X in IOV mode — Selected when `GPIE.MULTIPLE_MSIX` is set (as previously stated) and `GPIE.VT_MODE` DOES NOT equal 00b.

The following sections describe the interrupt registers and device functionality at all operation modes.

7.3.1 Interrupt Registers

7.3.1.1 Physical Function (PF) Registers

The PF interrupt logic consists of the registers listed in the [Table 7-46](#) followed by their description:

Table 7-46. PF Interrupt Registers

Acronym	Complete Name
EICR	Extended Interrupt Cause register
EICS	Extended Interrupt Cause Set register (enables software to initiate interrupts)
EIMS	Extended Interrupt Mask Set/Read register
EIMC	Extended Interrupt Mask Clear register
EIAC	Extended Interrupt Auto Clear register (following interrupt assertion)
EIAM	Extended Interrupt Auto Mask register (auto set/clear of the EIMS)
EITR	Extended Interrupt Throttling register (throttling)
IVAR	Interrupt Vector Allocation Registers (described in Section 7.3.4)
IVAR_MISC	Miscellaneous Interrupt Vector Allocation Register (described in Section 7.3.4)

These registers are extended to 64 bits by an additional set of two registers. EICR has an additional two registers EICR(1)... EICR(2) and so on for the EICS, EIMS, EIMC, EIAM and EITR registers. The EIAC register is not extended to 64 bits as this extended interrupt causes are always auto cleared. Any reference to EICR... EIAM registers as well as any global interrupt settings in the GPIE register relates to their extended size of 64 bits.

The legacy EICR[15:0] mirror the content of EICR(1)[15:0]. In the same manner the lower 16 bits of EICS, EIMS, EIMC, EIAM mirror the lower 16 bits of EICS(1), EIMS(1), EIMC(1), EIAM(1). For more details on the use of these registers in the various interrupt modes (legacy, MSI, MSI-X) see [Section 7.3.4](#).

7.3.1.2 Virtual Function (VF) Registers

The VF interrupt logic has the same set of interrupt registers while each of them has three entries for three interrupt causes. The names and functionality of these registers are the same as those of the PF with a prefix of VT as follows: VFEICR, VFEICS, VFEIMS, VFEIMC, VFEIAM, VFEITR. The interrupt causes are always auto cleared. Although each VF can generate up to three interrupts, only the first two registers are capable of interrupt throttling and are associated to VFEITR registers (see [Section 7.3.4.3.2](#) for its proper usage). Each VF also has the mapping registers VFIVAR and VFIVAR_MISC. Note that any global interrupt setting by the GPIE register affect both interrupt settings of the PF as well as the VFs.

7.3.1.3 Extended Interrupt Cause (EICR) Registers

This register records the interrupt causes to provide software information on the interrupt source. Each time an interrupt cause happens, the corresponding interrupt bit is set in the EICR registers. An interrupt is generated each time one of the bits in these registers is set, and the corresponding interrupt is enabled via the EIMS registers. The possible interrupt causes are as follows:

- Each *RTxQ* bit represents the following events: Tx or Rx descriptor write back; Rx queue full and Rx descriptor queue minimum threshold.
- Interrupts can be throttled by ITR as configured in the EITR register.
- Mapping the Tx and Rx queues to EICR is done by the IVAR registers as described in [Section 7.3.4](#). Each bit might represent an event on a single Tx or Rx queue or could represent multiple queues according to the IVAR setting. In the later case, software might not be able to distinguish between the interrupt causes other than checking all associated Tx and Rx queues.
- The *MULTIPLE_MSIX* = 1b setting is useful when multiple MSI-X vectors are assigned to the device. When the *GPIE.MULTIPLE_MSIX* bit is set, the *RTxQ* bits are associated with dedicated MSI-X vectors. Bit 0 is Tx/Rx interrupt associated with MSI-X vector 0 and bit 15 is Tx/Rx interrupt associated with MSI-X vector 15.

Writing a 1b to any bit in the register clears it. Writing a 0b to any bit has no effect. The EICR is also cleared on read if *GPIE.OCD* bit is cleared. When the *GPIE.OCD* bit is set, only bits 16...29 are cleared on read. The later setting is useful for MSI-X mode in which the Tx and Rx and possibly the timer interrupts do not share the same interrupt with the other causes. Bits in the register can be auto cleared depending on the EIAC register setting (see [Section 7.3.1.6](#)).

7.3.1.4 Extended Interrupt Cause Set (EICS) Register

This register enables software to initiate a hardware interrupt. Setting any bit on the EICS sets its corresponding bit in the EICR register while bits written to 0b have no impact. It then causes an interrupt assertion if enabled by the EIMS register. Setting any bit generates throttled interrupt depending on the *GPIE.EIMEN* setting: When the *EIMEN* bit is set, setting the EICS register causes an LLI interrupt; When the *EIMEN* bit is cleared, setting the EICS register causes an interrupt after the corresponding interrupt throttling timer expires.

Note: The *EIMEN* bit can be set high only when working in auto-mask mode (*EIAM* bit of the associated interrupt is set).

7.3.1.5 Extended Interrupt Mask Set and Read (EIMS) Register, and Extended Interrupt Mask Clear (EIMC) Register

The Extended Interrupt Mask Set and Read (EIMS) register enables the interrupts in the EICR. When set to 1b, each bit in the EIMS register, enables its corresponding bit in the EICR. Software might enable each interrupt by setting bits in the EIMS register to 1b. Reading EIMS returns its value. Software might clear any bit in the EIMS register by setting its corresponding bit in the Extended Interrupt Mask Clear (EIMC) register. Reading the EIMC register does not return any meaningful data.

This independent mechanism of setting and clearing bits in the EIMS register saves the need for read modify write and also enables simple programming in multi-thread, multi-CPU core systems.

Note: The EICR register stores the interrupt events regardless of the state of the EIMS register.

7.3.1.6 Extended Interrupt Auto Clear Enable (EIAC) Register

Each bit in this register enables auto clearing of its corresponding bit in EICR following interrupt assertion. It is useful for Tx and Rx interrupt causes that have dedicated MSI-X vectors. When the Tx and Rx interrupt causes share an interrupt with the other or a timer interrupt, the relevant EIAC bits should not be set. Bits in the EICR register that are not enabled by auto clear, must be cleared by either writing a 1b to clear or a read to clear.

Note that there are no EIAC(1)...EIAC(2) registers. The hardware setting for interrupts 16...63 is always auto clear.

Note: Bits 29:16 should never be set to auto clear since they share the same MSI-X vector.

Writing to the EIAC register changes the setting of the entire register. In IOV mode, some of the bits in this register might affect VF functionality (VF-56...VF-63). It is recommended that software set the register in PF before VF's are enabled. Otherwise, a software semaphore might be required between the VF and the PF to avoid setting corruption.

7.3.1.7 Extended Interrupt Auto Mask Enable (EIAM) Register

Each bit in this register enables auto clearing and auto setting of its corresponding bit in the EIMS register as follows:

- Following a write of 1b to any bit in the EICS register (interrupt cause set), its corresponding bit in the EIMS register is auto set as well enabling its interrupt.
- A write to clear the EICR register clears its corresponding bits in the EIMS register masking further interrupts.
- A read to clear the EICR register, clears the *EIMS* bits (enabled by the EIAM) masking further interrupts. Note that if the *GPIE.OCD* bit is set, Tx and Rx interrupt causes are not cleared on read (bits 0:15 in the EICR). In this case, bits 0:15 in the EIMS are not cleared as well.
- In MSI-X mode the auto clear functionality can be driven by MSI-X vector assertion if *GPIE.EIAME* is set.

Note: Bits 29:16 should never be set to auto clear since they share the same MSI-X vector.

Writing to the EIAM register changes the setting of the entire register. In IOV mode, some of the bits in this register might affect VF functionality. It is recommended that software set the register in PF before VF's are enabled. Otherwise, a software semaphore might be required between the VF and the PF to avoid setting corruption.

If any of the *Auto Mask Enable* bits is set in the EIAM registers, the GPIE.EIAME bit must be set as well.

7.3.2 Interrupt Moderation

Interrupt rates can be tuned by the EITR register for reduced CPU utilization while minimizing CPU latency. In MSI or legacy interrupt modes, only EITR register 0 can be used. In MSI-X, non-IOV mode, the E610 includes 64 EITR registers 0...63 that are mapped to MSI-X vectors 0...63, respectively. In IOV mode, there are an additional 65 EITR registers that are mapped to the MSI-X vectors of the virtual functions. The mapping of MSI-X vectors to EITR registers are described in [Section 7.3.1.3](#).

7.3.2.1 Time-Based Interrupt Throttling — ITR

Time-based interrupt throttling is useful to limit the maximum interrupt rate regardless of network traffic conditions. The ITR logic is targeted for Rx/Tx interrupts only. It is assumed that the software device driver does not moderate the timer, other and mail box (IOV mode) interrupts. In non-IOV mode, all 64 interrupts can be associated with ITR logic. In IOV mode, the ITR logic is shared between the PF and VFs as shown in [Figure 7-22](#). The ITR mechanism is based on the following parameters:

- **ITR Interval** field in the EITR registers — The minimum inter-interrupt interval is specified in 2.048 μ s units (at 1 Gb/s or 10 Gb/s link). When the ITR Interval equals zero, interrupt throttling is disabled and any event causes an immediate interrupt. The field is composed of nine bits enabling a range of 2.048 μ s up to 1046.528 μ s. These ITR interval times correspond to interrupt rates in the range of 488 K INT/sec to 955 INT/sec. When operating at 100 Mb/s link, the ITR interval is specified in 20.48 μ s units.
 - Due to internal synchronization issues, the ITR interval can be shortened by up to 1 μ s at 10 Gb/s or 1 Gb/s link and up to 10 μ s at 100 Mb/s link when it is triggered by packet write back or interrupt enablement.
- **ITR Counter** partially exposed in the EITR registers — Down counter that is loaded by the ITR interval each time the associated interrupt is asserted.
 - The counter is decremented by one each 1.024 μ s (at 1 Gb/s or 10 Gb/s link) and stops decrementing at zero. At 100 Mb/s link, the speed of the counter is decremented by one each 10.24 μ s.
 - If an event happens before the counter is zero, it sets the EICR. The interrupt can be asserted only when the ITR time expires (counter is zero).
 - Else (no events during the entire ITR interval), the EICR register is not set and the interrupt is not asserted on ITR expiration. The next event sets the EICR bit and generates an immediate interrupt.
 - Once the interrupt is asserted, the ITR counter is loaded by the ITR interval and the entire cycle re-starts. The next interrupt can be generated only after the ITR counter expires once again.

7.3.2.2 Immediate Interrupt

The E610 might initiate an immediate interrupt when the receive descriptor ring is almost empty (Rx descriptors below a specific threshold). The threshold is defined by `SRRCTL[n].RDMTS` per Rx queue. This mechanism can protect against memory resources being used up during reception of a long burst of short packets.

7.3.3 TCP Timer Interrupt

7.3.3.1 Introduction

To implement TCP timers, software needs to take action periodically (every 10 ms). Today, the driver must rely on software-based timers, whose granularity can change from platform to platform. This software timer generates a software NIC interrupt, which then enables the driver to perform timer functions, avoiding cache thrash and enabling parallelism. The timer interval is system-specific.

It would be more accurate and more efficient for this periodic timer to be implemented in hardware. The driver would program a timeout value (usual value of 10 ms), and each time the timer expires, hardware sets a specific bit in the EICR register. When an interrupt occurs (due to normal interrupt moderation schemes), software reads the EICR register and discovers that it needs to process timer events.

The timeout should be programmable by the driver, and the driver should be able to disable the timer interrupt if it is not needed.

7.3.3.2 Description

A stand-alone, down-counter is implemented. An interrupt is issued each time the value of the counter is zero.

Software is responsible for setting an initial value for the timer in the `TCPTIMER.Duration` field. Kick-starting is done by writing a 1b to the `TCPTIMER.KickStart` bit.

Following kick starting, an internal counter is set to the value defined by the `Duration` field. Then the counter is decreased by one each ms. When the counter reaches zero, an interrupt is issued. The counter re-starts counting from its initial value if the `TCPTIMER.Loop` field is set.

7.3.4 Mapping of Interrupt Causes

The following sections describe legacy, MSI and MSI-X interrupt modes.

7.3.4.1 Legacy and MSI Interrupt Modes

In legacy and MSI modes, an interrupt cause is reflected by setting one of the bits in the EICR register, where each bit reflects one or more causes. All interrupt causes are mapped to a single interrupt signal: either legacy INTA/B or MSI. This section describes the mapping of interrupt causes (that is a specific Rx or Tx queue event or any other event) to bits in the EICR.

The TCP timer and all other interrupt causes are mapped directly to `EICR[30:16]`. Note that the `IVAR_MISC` register is not used in legacy and MSI modes.

Mapping the Tx and Rx queues to interrupt bits in the EICR register is programmed in the `IVAR` registers as shown in [Figure 7-22](#). Each entry in the `IVAR` registers is composed of two fields that identify the associated bit in the `EICR[15:0]` register. Software might map multiple Tx and Rx queues to the same EICR bit.

- **INT_Alloc** — Defines one of the bits (0...15) in the EICR register that reflects the interrupt status indication.
- **INT_Alloc_val** — Valid bit for the this interrupt cause.

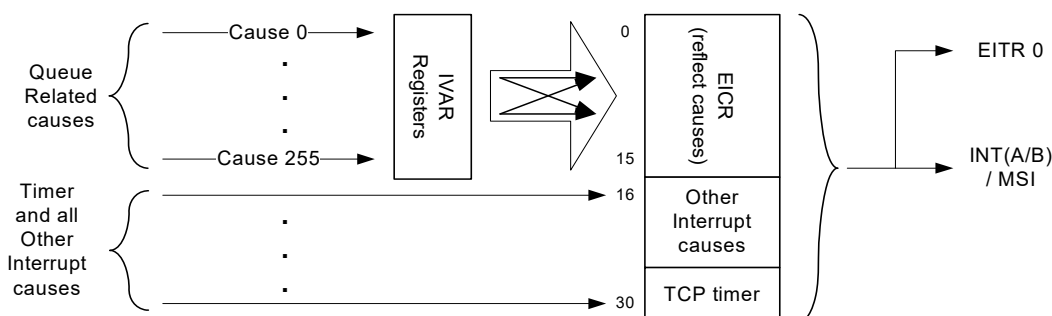


Figure 7-22. Cause Mapping in Legacy and MSI Modes

Mapping between the Tx and Rx queue to the IVAR registers is hard-wired as shown in the [Figure 7-23](#).

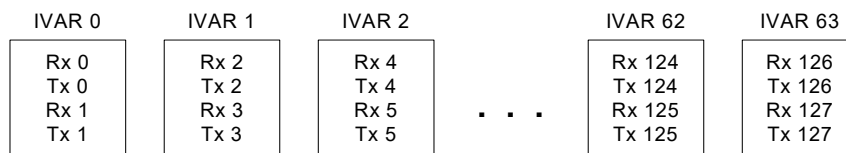


Figure 7-23. Rx and Tx Queue Mapping to IVAR Registers

7.3.4.2 MSI-X Mode in Non-IOV Mode

- MSI-X defines a separate optional extension to basic MSI functionality. Hardware indicates the number of requested MSI-X vectors in the table size in the MSI-X capability structure in the configuration space. The number of requested MSI-X vectors is loaded from NVM in the `PCI_CNFG2.MSI_X_PF_N` field up to maximum of 64 MSI-X vectors. The operating system might allocate any number of MSI-X vectors to the device from a minimum of one up to the requested number of MSI-X vectors.
- Enables interrupts causes allocation to the assigned MSI-X vectors. Interrupt allocation is programmed by the IVAR registers and are described in this section.
- Each vector can use an independent address and data value as programmed directly by the operating system in the MSI-X vector table.
- Each MSI-X vector is associated to an EITR register with the same index (MSI-X 0 to EITR[0], MSI-X 1 to EITR[1],...).

For more information on MSI-X, refer to the PCI Local Bus Specification, Revision 3.0.

MSI-X vectors can be used for several purposes:

1. Dedicated MSI-X vectors per interrupt cause (avoids the need to read the interrupt cause register).
2. Load balancing by MSI-X vectors assignment to different CPUs.
3. Optimized interrupt moderation schemes per MSI-X vector using the EITR registers.

The MSI-X vectors are used for Tx and Rx interrupt causes as well as the other and timer interrupt causes. The remainder of this section describes the mapping of interrupt causes (such as a specific Rx or Tx queue event or any other event) to the interrupts registers and the MSI-X vectors.

The TCP timer and other events are reflected in EICR[30:16] the same as the legacy and MSI mode. It is then mapped to the MSI-X vectors by the IVAR_MISC register as shown in Figure 7-24. The IVAR_MISC register includes two entries for the timer interrupt and an additional entry for all the other causes. The structure of each entry is as follows:

- **INT_Alloc** — Defines the MSI-X vector (0...63) assigned to this interrupt cause.
- **INT_Alloc_val** — Valid bit for the this interrupt cause.

The Tx and Rx queues are associated to the IVAR0...IVAR63 the same as legacy and MSI mode shown in Figure 7-23. The Tx and Rx queues are mapped by the IVAR registers to EICR(1)...EICR(2) registers and MSI-X vectors 0...63 illustrated in Figure 7-24. The IVAR entries have the same structure as the IVAR_MISC register previously shown. Each bit in EICR(1...2) registers is associated to MSI-X vector 0...63 as follows:

- EICR(i).bit_num is associated to MSI-X vector ($i \times 32 + \text{bit\_num}$).
- The legacy EICR[15:0] mirror the content of EICR(1)[15:0]. In the same manner the lower 16 bits of EICS, EIMS, EIMC, EIAM mirror the lower 16 bits of EICS(1), EIMS(1), EIMC(1), EIAM(1). The use of these registers depends on the number of assigned MSI-X interrupts as follows:
- **16 Tx and Rx Interrupts** — When using up to 16 Tx and Rx interrupts, software might access the Tx and Rx interrupt bits in the legacy EICR, EICS,... registers.
- **More than 16 Tx and Rx Interrupts** — When using more than 16 Tx and Rx interrupts, software must use EICS(1)...EICS(2), EIMS(1)...EIMS(2),... In the later case, software should avoid modifying the lower 16 bits in the EICS, EIMS... registers when it accesses the higher bits of these registers as follows:
 - EICR, EICS, EIMS and EIMC — When software programs the higher 16 bits of these registers, it should set their lower 16 bits to zero's keeping the EICR(1), EICS(1), EIMS(1) and EIMC(1) unaffected.
 - EIAM — When software programs the higher 16 bits, it should keep the lower 16 bits at their previous setting so the EIAM(1) is unaffected.
 - EIAC — When software programs the higher 16 bits, it should set the lower 16 bits to one's.
- **Single MSI-X vector** — If the operating system allocates only a single MSI-X vector, the driver might use the non-MSI-X mapping method (setting the GPIE.Multiple_MSIX to 0b). In this case, the *INT_Alloc* field in the IVAR registers might define one of the lower 16 bits in the EICR register while using MSI-X vector 0. The IVAR_MISC should be programmed to MSI-X vector 0.

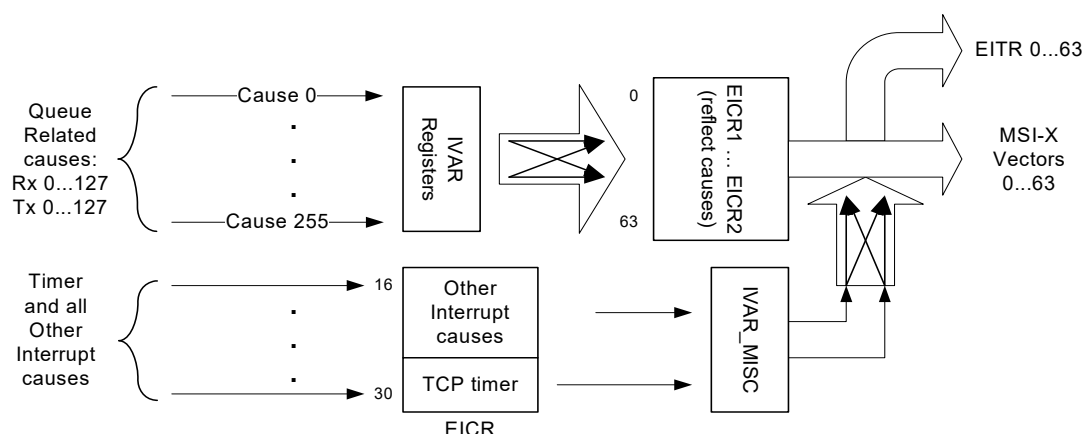


Figure 7-24. Cause Mapping in MSI-X Mode (non-IOV)

7.3.4.3 MSI-X Interrupts in IOV Mode

In IOV mode, interrupts must be implemented by MSI-X vectors. The E610 supports up to 64 virtual functions VF(0...63). Each VF can generate up to three MSI-X vectors. The number of requested MSI-X vectors per VF is loaded from NVM in the `PCI_CNFG2.MSI_X_VF_N` field. It is reflected in the *Table Size* field in the PCIe MSI-X capability structure of the VF's. In addition, the PF requires its own interrupts. The number of requested MSI-X vectors is loaded from NVM in the `PCI_CNFG2.MSI_X_PF_N` field up to maximum of 64 MSI-X vectors. It is reflected in the *Table Size* field in the PCIe MSI-X capability structure.

7.3.4.3.1 MSI-X Vectors Used by Physical Function (PF)

PF is responsible for the timer and other interrupt causes that include the VM to PF mailbox cause (explained in the virtualization sections). These events are reflected in `EICR[30:16]` and MSI-X vectors are the same as the non-IOV mode (illustrated in [Figure 7-22](#)). When there are less than the maximum possible active VF's, some of the Tx and Rx queues can be associated with the PF. These queues can be used for the sake of additional VM's serviced by the hypervisor (the same as VMDq mode) or some Kernel applications handled by the hypervisor. Tx and Rx mapping to the IVAR registers is shown in [Figure 7-23](#) and mapping to the `EICR`, `EICR(1)`,...`EICR(2)` registers as well as the MSI-X vectors is shown in [Figure 7-24](#). See [Section 7.3.4.3.3](#) for MSI-X vectors mapping of PF and VF's to the `EITR` registers.

Note: Software should not assign MSI-X vectors in the PF to Tx and Rx queues that are assigned to other VF's. In the case that VF's become active after the PF used the relevant Tx and Rx queues, it is the responsibility of the PF driver to clear all pending interrupts of the associated MSI-X vectors.

7.3.4.3.2 MSI-X Vectors Used by Virtual Functions (VFs)

Each of the VFs in IOV mode is allocated separate IVAR(s) called VFIVAR registers, and a separate IVAR_MISC called VFIVAR_MISC register. The VFIVAR_MISC maps the mailbox interrupt of the VF to its VFEICR and the MSI-X vector. The VFIVAR registers map the Tx and Rx interrupts of the VF to its VFEICR and the MSI-X vector. The mapping is similar to the mapping in the PF as shown in [Figure 7-25](#) with the following comments:

- Each VF cannot have more than three MSI-X vectors. It has only three active bits in the VFEICR register while `VFEICR.bit_num` is associated with MSI-X vector (`bit_num`).
- The Tx and Rx interrupt can be mapped only to MSI-X 0 and MSI-X 1 (associated with `VFEICR.0` and `VFEICR.1`).
- The mailbox interrupt can be mapped to any of the three MSI-X vectors. However, when all three of them are allocated by the operating system, software should map the mailbox to MSI-X 2 (associated with `VFEICR.2`). This rule should be kept since only `VFEICR.0` and `VFEICR.1` have ITR registers (`VFEITR-0` and `VFEITR-1`).
- Association between the Tx and Rx queues and the VFIVAR registers is shown in the [Figure 7-25](#), [Figure 7-26](#) and [Figure 7-27](#) for IOV-64 (64 VF's), IOV-32 and IOV-16. The colored boxes in the figures show the mapping between VF Rx and Tx queues to VFIVAR registers while the dashed boxes show the physical IVAR registers and the associated physical Rx and Tx queues.

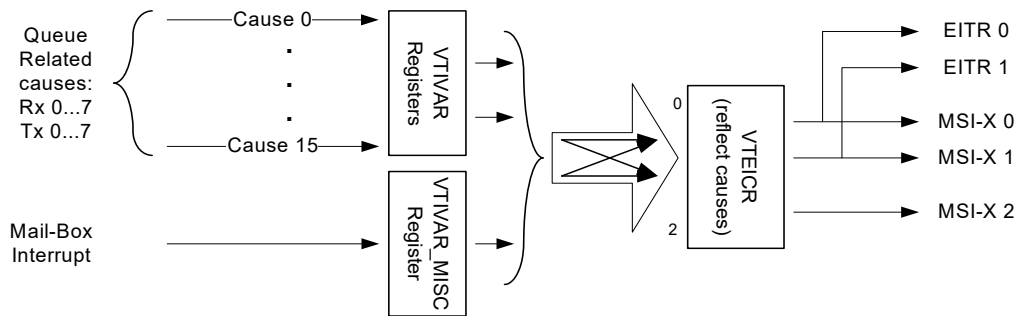


Figure 7-25. VF Interrupt Cause Mapping (MSI-X, IOV)

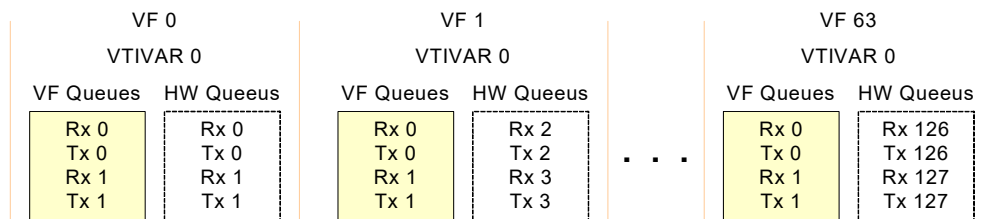


Figure 7-26. VF Mapping of Rx and Tx Queue to VFIVAR in 64 VF's Mode

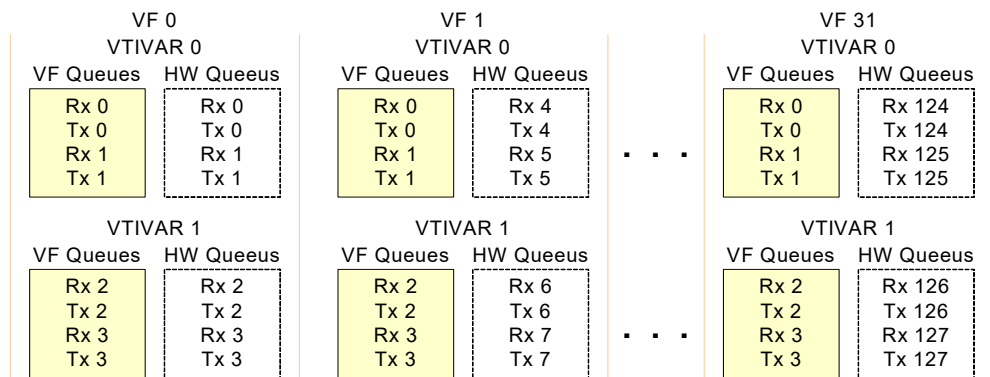


Figure 7-27. VF Mapping of Rx and Tx Queue to VFIVAR in 32 VF's Mode

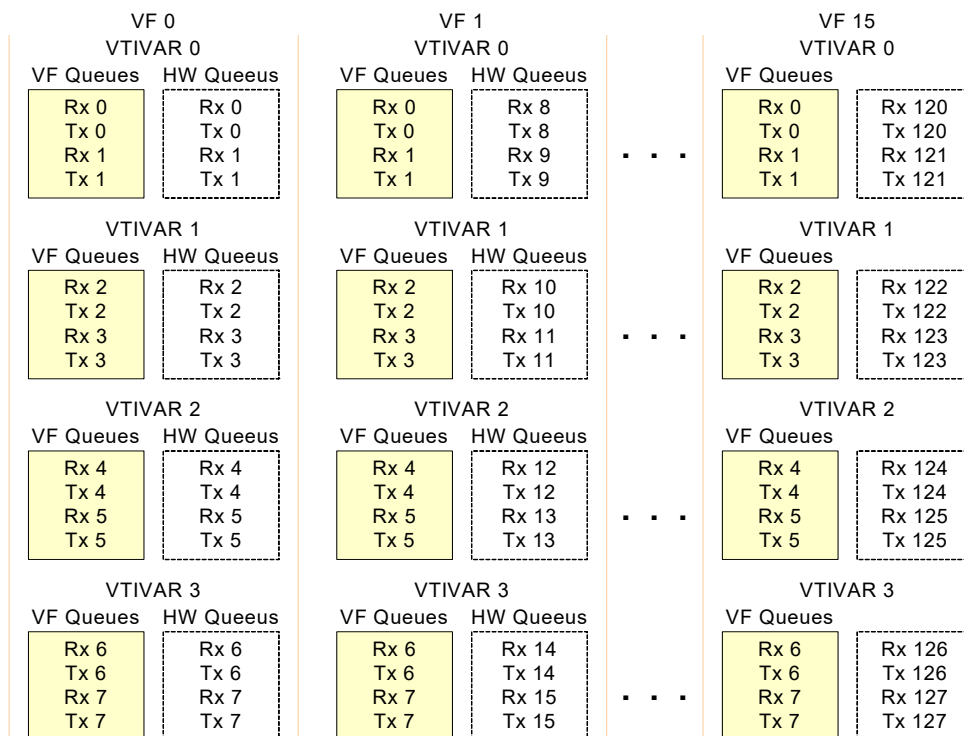
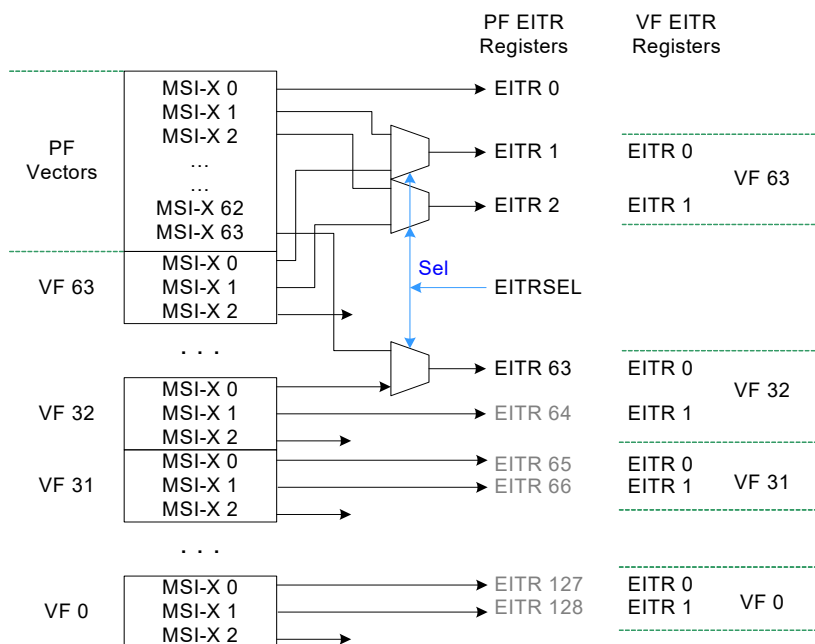


Figure 7-28. VF Mapping of Rx and Tx Queue to VFIVAR in 16 VF's Mode

7.3.4.3.3 MSI-X Vectors Mapping to EITR

EITR registers are aimed for Tx and Rx interrupt throttling. In IOV mode, the Tx and Rx queues might belong to either the PF or to the VF's. EITR(1...63) are multiplexed between the PF and the VF's as configured by the EITRSEL register. Figure 7-29 and Table 7-47 show the multiplexing logic and required software settings. For any active VF (starting from VF32 and above), software should program the matching bit in the EITRSEL to 1b. For any EITR that belongs to a VF, software should not map any interrupt causes in the PF to an MSI-X vector that is associated with the same EITR register.



MSI-X 2 on each VF has no associated EITR register. It is useful for the mailbox interrupts that do not require interrupt moderation.

Figure 7-29. PF/VF MSI-X Vectors Mapping to EITR

Table 7-47. PF/VF MSI-X Vectors Mapping Table to EITR Registers

VM Active	EITRSEL.N Setting	MSI-X Routing to EITR
Non-IOV or VF(32...63) inactive	EITRSEL must be set to 0x0000	MSI-X(1...63) -> EITR(1...63)
VF(32) active	EITRSEL[0] must be set to 1b	VF(32) MSI-X(0) -> EITR(63)
VF(33) active	EITRSEL[1] must be set to 1b	VF(33) MSI-X(1) -> EITR(62) VF(33) MSI-X(0) -> EITR(61)
VF(34) active	EITRSEL[2] must be set to 1b	VF(34) MSI-X(1) -> EITR(60) VF(34) MSI-X(0) -> EITR(59)
...	...	...
VF(62) active	EITRSEL[30] must be set to 1b	VF(62) MSI-X(1) -> EITR(4) VF(62) MSI-X(0) -> EITR(3)
VF(63) active	EITRSEL[31] must be set to 1b	VF(63) MSI-X(1) -> EITR(2) VF(63) MSI-X(0) -> EITR(1)

7.4 802.1q VLAN Support

The E610 provides several specific mechanisms to support 802.1q VLANs:

- Optional adding (for transmits) and stripping (for receives) of IEEE 802.1q VLAN tags.
- Optional ability to filter packets belonging to certain 802.1q VLANs.

7.4.1 802.1q VLAN Packet Format

The following table compares an untagged 802.3 Ethernet packet with an 802.1q VLAN tagged packet:

802.3 Packet	#Octets	802.1q VLAN Packet	#Octets
DA	6	DA	6
SA	6	SA	6
Type/Length	2	802.1q Tag	4
Data	46-1500	Type/Length	2
CRC	4	Data	46-1500
		CRC*	4

Note: The CRC for the 802.1q tagged frame is re-computed, so that it covers the entire tagged frame including the 802.1q tag header. Also, maximum frame size for an 802.1q VLAN packet is 1522 octets as opposed to 1518 octets for a normal 802.3z Ethernet packet.

7.4.2 802.1q Tagged Frames

For 802.1q, the *Tag Header* field consists of four octets comprised of the Tag Protocol Identifier (TPID) and Tag Control Information (TCI); each taking two octets. The first 16 bits of the tag header makes up the TPID. It contains the protocol type that identifies the packet as a valid 802.1q tagged packet.

The two octets making up the TCI contain three fields as follows:

- User Priority (UP)
- Drop Eligible Indicator (DEI) — Should be set to 0b for transmits. For receives, the device has the capability to filter out packets that have this bit set. See the *DEIEN* and *DEI* bits in the *VLNCTRL*.
- VLAN Identifier (VID)

Octet 1		Octet 2
UP	DEI	VID

7.4.3 Transmitting and Receiving 802.1q Packets

Since the 802.1q tag is only four bytes, adding and stripping of tags can be done completely in software. In other words, for transmits, software inserts the tag into packet data before it builds the transmit descriptor list, and for receives, software strips the 4-byte tag from the packet data before delivering the packet to upper layer software. However, because adding and stripping of tags in software adds overhead for the host, the E610 has additional capabilities to add and strip tags in hardware. See [Section 7.4.3.1](#) and [Section 7.4.3.2](#).

7.4.3.1 Adding 802.1q Tags on Transmits

The inner VLAN header can be added by software in one of the following methods:

- The header is included in the transmit data buffers.
- Software might instruct the E610 to insert an 802.1q VLAN tag on a per-packet basis. If the *VLE* bit in the transmit descriptor is set to 1b, the E610 inserts a VLAN tag into the packet that it transmits over the wire. The Tag Protocol Identifier — TPID (VLAN Ether Type) field of the 802.1q tag comes from the *DMATXCTL.VT*, and the Tag Control Information (TCI) of the 802.1q tag comes from the VLAN field of the legacy transmit descriptor or the *VLAN Tag* field of the advanced data transmit descriptor.
- In IOV mode, the priority tag, DEI and VLAN ID can be taken from the *PFVMVIR* (see details in [Section 7.8.4.2](#))

7.4.3.2 Stripping 802.1q Tags on Receives

Software might instruct the E610 to strip 802.1q VLAN tags from received packets. The policy whether to strip the VLAN tag is configurable per queue.

If the *RXDCTL.VME* bit for a given queue is set to 1b, and the incoming packet is an 802.1q VLAN packet (that is, its *Ethernet Type* field matched the *VLNCTRL.VET*), the E610 strips the 4-byte VLAN tag from the packet, and stores the TCI in the *VLAN Tag* field of the receive descriptor.

The E610 also sets the *VP* bit in the receive descriptor to indicate that the packet had a VLAN tag that was stripped. If the *RXDCTL.VME* bit is not set, the 802.1q packets can still be received if they pass the receive filter, but the VLAN tag is not stripped and the *VP* bit is not set. If *PFQDE.HIDE_VLAN* is set, the VLAN tag is stripped as above, but the *VLAN Tag* field and the *STATUS.VP* bit in the Rx descriptor are cleared.

7.4.4 802.1q VLAN Packet Filtering

VLAN filtering is enabled by setting the *VLNCTRL.VFE* bit to 1b. If enabled, hardware compares the *Type* field of the incoming packet to a 16-bit field in the VLAN Ether Type (VET) register. If the *VLAN Type* field in the incoming packet matches the VET register, the packet is compared against the VLAN Filter Table Array for acceptance.

The VLAN filter register VTFA, is a vector array composed of 4096 bits. The VLAN ID (VID) is a 12-bit field in the VLAN tag that is used as an index pointer to this vector. If the VID in a received packet points to an active bit (set to 1b), the packet matches the VLAN filter. The 4096-bit vector is comprised of 128 x 32 bit registers. The upper 7 bits of the VID selects one of the 128 registers while the lower 5 bits map the bit within the selected register.

Two other bits in the VLNCTRL register, *DEIEN* and *DEI*, are also used in conjunction with 802.1q VLAN filtering operations. *DEIEN* enables the comparison of the value of the *DEI* bit in the 802.1q packet to the Receive Control register *DEI* bit as acceptance criteria for the packet.

Note: The *VFE* bit does not effect whether the VLAN tag is stripped. It only effects whether the VLAN packet passes the receive filter.

7.4.5 Double VLAN and Single VLAN Support

The E610 supports a mode where all received and sent packets have at least one VLAN tag in addition to the regular tagging that might optionally be added. In this document, when a packet carries two VLAN headers, the first header is referred to as an outer VLAN and the second header as an inner VLAN header (as listed in the table that follows). This mode is used for systems where the near end switch adds the outer VLAN header containing switching information. This mode is enabled by the following configuration:

- This mode is activated by setting the *DMATXCTL.GDV* and the *EXTENDED_VLAN* bit in the *CTRL_EXT* register.
- The EtherType of the VLAN tag used for the additional VLAN is defined in the *VET_EXT* field in the *EXVET* and *EXVET_T* registers.

7.4.5.1 Cross Functionality with Manageability

The E610 does not provide any stripping or adding VLAN header(s) to manageability packets. Therefore, packets that are directed to/from the manageability controller should include the VLAN headers as part of the Rx/Tx data. The manageability controller should know if the E610 is set to double VLAN mode as well as the VLAN EtherType(s).

Table 7-48. Double VLAN Packet Format

MAC Address	Outer VLAN	Inner VLAN	L2 Payload	Ethernet CRC

7.4.5.2 Transmit Functionality

7.4.5.2.1 Transmit Functionality on the Outer VLAN Header

- A packet with a single VLAN header is assumed to have only the outer VLAN.
- The outer VLAN header must be added by software as part of the Tx data buffers.
- Hardware does not relate to the outer VLAN header other than the capability of skipping it for parsing inner fields.
- Hardware expects that any transmitted packet (see the disclaimer that follows) has at least the outer VLAN added by software. For any offload that hardware might provide in the transmit data path, hardware assumes that the outer VLAN is present. For those packets that an outer VLAN is not present, any offload that relates to inner fields to the EtherType might not be provided.

7.4.5.2.2 Transmit Functionality on the Inner VLAN Header

- Inner VLAN insertion is handled as described in [Section 7.4.3.1](#).
- Hardware identifies and skips the VLAN header for parsing inner fields.
- DCB — The traffic class is dictated by the Tx queue.
- Pool Filtering — Destination pool(s) and anti-spoofing functionality is based on the Ethernet MAC Address and inner VLAN (if present) as described in [Section 7.8.3.4](#) and [Section 7.8.4.2](#).

7.4.5.3 Receive Handling of Packets with VLAN Header(s)

A received frame is analyzed for the existence of the outer and inner VLAN headers. The procedure is as follows:

```
Check the EtherType against the outer VLAN ID. If match then
{
    Check the next EtherType against the inner VLAN ID. If match
    {
        This is the double VLAN case. Process both outer and inner VLANs as described below
    }
    Else
    {
        Only an outer VLAN exists. Process the outer as described below. Assume no inner VLAN
    }
}
Else
{
    Check the EtherType against the inner VLAN ID. If match
    {
        This is the case of an inner VLAN only. Handle the frame as an unknown frame - device
        does not provide any offloads
    }
    Else
    {
        This is the case of no VLAN. Process the frame (ignore outer and inner VLAN processing)
    }
}
```

7.4.5.3.1 Receive Functionality on the Outer VLAN Header

- Hardware checks the EtherType of the outer VLAN header against the programmed value in the EXVET register. VLAN header presence is indicated in the STATUS.VEXT bit in the Rx descriptor.
- The outer VLAN header is posted as is to the receive data buffers.

7.4.5.3.2 Receive Functionality on the Inner VLAN Header

- Hardware checks the EtherType of the inner VLAN header against the programmed value in the VLNCTRL.VET. VLAN header presence is indicated in the STATUS.VP bit in the Rx descriptor.
- L2 packet filtering is based on the VLAN ID in the inner VLAN header.
- Pool Filtering — Destination pool(s) are defined by the Ethernet MAC Address and inner VLAN (if presence) as described in [Section 7.8.3.3](#).
- Inner VLAN tag striping is handled as described in [Section 7.4.3.2](#).

7.4.5.4 Packets with No VLAN Headers in Double VLAN Mode

There are some cases when packets might not carry any VLAN headers, even when extended VLAN is enabled. A few examples for packets that might not carry any VLAN header are: flow control and priority flow control, LACP, LLDP, GMRP, and optional 802.1x packets. When it is expected to transmit untagged packets by software in Double VLAN Mode the software must not enable VLAN anti-spoofing and VLAN validation nor transmit to receive switching.

7.4.5.4.1 Transmit Functionality

DCB — The TC in the Tx data path is directed by the Tx queue of the transmitted packet.

Transmit offload functionality — Software should not enable any offload functions.

7.4.5.4.2 Receive Functionality

Receive offload functionality — pool and queue are selected by the Ethernet MAC Address or ETQF/ETQS registers. Filtering to host and manageability remains functional.

7.4.5.5 Packets with Two VLAN Headers Not in Double VLAN Mode

When the *EXTENDED_VLAN* bit in the *CTRL_EXT* register and *DMATXCTL.GDV* are not set, hardware expects that Rx and Tx packets might not carry a VLAN header or a single VLAN header. Hardware does not relate to the programming of the *VET_EXT* field in the *EXVET* register. Tx and Rx handling of packets with double VLAN headers is unexpected.

7.4.6 E-tag and VLAN

In some systems an additional external tag (E-tag) can be present before the VLAN. This section describes the support for VLANs in presence of external tags. This mode is used for systems where the device adds a tag to identify a subsystem (usually a VM) and the near end switch adds a tag indicating the destination subsystem. External tags may be present on part of the packets and missing in others.

7.4.6.1 Transmit Functionality

7.4.6.1.1 Transmit Functionality on the External Tag

The E610 supports insertion of an external tag from a per pool register (PFVMTIR - [Section 8.2.2.20.16](#)) and a VLAN tag from a per pool register (PFVMVIR - [Section 8.2.2.20.17](#)) or from the descriptor as indicated by the *VLE* bit. The following options are supported:

(Inner) VLAN Source	(Outer) External Tag Source	
	Hardware Register - PFVMTIR/PFVMVIR	Embedded in Packet
Hardware register - PFVMVIR	Supported	Not supported
Descriptor	Supported	Not supported
Embedded in Packet	Supported	Supported

After the tags are inserted in the packet, the E610 uses the VLAN tag and the EtherType as part of the forwarding decision as described in [Section 7.8.3.4](#). In a packet with at most one external tag and one VLAN, the VLAN and EtherType are identified correctly.

7.4.6.2 Receive Handling of Packets with External Tags

External tags can be extracted from the packet according to the PFQDE.STRIP_TAG field ([Section 8.2.2.20.12](#)). Inner tag extraction is handled as described above ([Section 7.4.3.2](#)). User priority to TC mapping is described in [Section 7.4.6.4](#).

7.4.6.3 Cross Functionality with Manageability

The E610 does not provide any stripping or adding VLAN header(s) to manageability packets. Therefore, packets that are directed to/from the manageability controller should include the L2 headers as part of the Rx/Tx data. The manageability controller should know if it is expected to add/receive an external tag as part of the packet.

7.4.6.4 Packet User Priority (802.1P) Bits Handling

The user priority bits may be used by the device for multiple purposes:

1. Defining the traffic class to which the traffic is associated ([Section 7.6.3.1](#)).
2. Defining which packets get a Timestamp in Buffer ([Section 7.1.6.2](#)).

For all these purposes, the UP bits are expected from the outermost tag with UP bits (E-tag or VLAN). Software can control it by the VLNCTRL.UP_FIRST_TAG_EN flag as described in [Table 7-49](#).

Table 7-49. UP Extraction Rules

Packet Type	VLNCTRL.UP_FIRST_TAG_EN = 1b	VLNCTRL.UP_FIRST_TAG_EN = 0b
Un-tagged Packet	User priority = 0	User priority = 0
Packet with a single VLAN	The user priority field from the VLAN header.	Single VLAN mode: The user priority field from the VLAN header. Double VLAN mode: User priority = 0
Packet with a VLAN and an outer tag (outer VLAN or E-tag)	The user priority field from the outer VLAN header	The user priority field from the inner VLAN header.
Packet with E-tag only	The user priority field from the E-tag	User priority = 0

7.5 TLP Processing Hints (TPH)

The E610 supports the TPH capability defined in the PCI Express specification. It does not support Extended TPH requests.

On the PCIe link existence of a TLP Process Hint (TPH) is indicated by setting the *TH* bit in the TLP header. Using the PCIe TLP Steering Tag (ST) and Processing Hints (PH) fields, the E610 can provide hints to the root complex about the destination (socket ID) and about data access patterns (locality in Cache), when executing DMA memory writes or read operations. Supply of TLP Processing Hints facilitates optimized processing of transactions that target Memory Space.

To enable TPH usage:

1. For a given function, the *TPH Requester Enable* bit in the PCIe configuration TPH Requester Control Register should be set.
2. Appropriate *TPH Enable* bits in TPH_RXCTRL or TPH_TXCTRL registers should be set.
3. Processing hints should be programmed in the TPH_CTRL.DESC_PH and TPH_CTRL.DATA_PH Processing hints (PH) fields.
4. Steering information should be programmed in the CPUID fields in the TPH_RXCTRL and TPH_TXCTRL registers.

The Processing hints (PH) and Steering Tags (ST) are set according to the characteristics of the traffic as described in [Table 7-50](#).

Note: To enable TPH usage, all the memory reads are done without setting any of the byte enable bits.

Note: Per queue, the TPH features are exclusive. Software can enable the TPH feature for a given queue.

7.5.1 Steering Tag and Processing Hint Programming

[Table 7-50](#) describes how the Steering tag (socket ID) and Processing hints are generated and how TPH operation is enabled for different types of DMA traffic.

Table 7-50. Steering Tag and Processing Hint Programming

Traffic type	ST Programming	PH value	Enable
Transmit descriptor write back or head write back	TPH_TXCTRL.CPUID ¹	TPH_CTRL.DESC_PH ²	<i>Tx Descriptor Writeback TPH EN</i> field in TPH_TXCTRL.
Receive data buffers write	TPH_RXCTRL.CPUID ¹	TPH_CTRL.DATA_PH ³	<i>Rx Header TPH EN</i> or <i>Rx Payload TPH EN</i> fields in TPH_RXCTRL.
Receive descriptor writeback	TPH_RXCTRL.CPUID ¹	TPH_CTRL.DESC_PH ²	<i>Rx Descriptor Writeback TPH EN</i> field in TPH_RXCTRL.
Transmit descriptor fetch	TPH_TXCTRL.CPUID ⁴	TPH_CTRL.DESC_PH ²	<i>Tx Descriptor Fetch TPH EN</i> field in TPH_TXCTRL.
Receive descriptor fetch	TPH_RXCTRL.CPUID ²	TPH_CTRL.DESC_PH ²	<i>Rx Descriptor fetch TPH EN</i> field in TPH_RXCTRL.
Transmit packet read	TPH_TXCTRL.CPUID ²	TPH_CTRL.DATA_PH ³	<i>Tx Packet TPH EN</i> field in TPH_TXCTRL.

1. The driver should always set bits [7:3] to zero and place Socket ID in bits [2:0].
2. Default is 00b (Bidirectional data structure).
3. Default is 10b (Target).
4. The hints are always zero.

7.6 Data Center Bridging (DCB)

See [Section 4.6.10](#) for the DCB configuration sequence.

7.6.1 Overview

DCB is a set of features that improve the capability of Ethernet to handle multiple traffic types (such as LAN, storage, IPC) by answering the various needs of those types. DCB enables multiple traffic types that have different requirements of packet delivery, bandwidth allocation and delay. Each traffic type can have one or several user priorities, or Traffic Classes (TCs). For example, IPC might have a high priority class for synchronization messages between servers and lower priority traffic class for bulk traffic exchange between servers. Most of the DCB functions impact the transmit traffic generated from the end node to the network (traffic generation). The receive data path needs be compliant with the requirements of DCB and provide the required functions as a traffic termination point.

DCB system requirements include:

- **Bandwidth grouping** — For effective multiplexing that simulates a separate link for the separate types of traffic, DCB requires that traffic types be recognized as groups in the bandwidth and priority handling by nodes in the network. Traffic types are associated to Bandwidth Groups (BWGs). The system needs to be able to allocate bandwidth to the BWGs in a way that emulates that group being on its own separate link.
- **Bandwidth fairness** — DCB multiplexing functions (transmit) and de-multiplexing functions (receive) need to guarantee minimum allocation of bandwidth to traffic types and traffic classes. Fairness between groups comes first, then fairness between TCs. If system resources (such as PCIe bandwidth) limit total throughput, then the available bandwidth should be distributed among consumers proportionally to their allocations.
- **Latency of operation** — DCB multiplexing and de-multiplexing functions need to allow minimum latency for some TCs. Arbitration mechanisms, packet buffers, descriptor queues and flow control algorithm need to be defined and designed to allow this. The best example is the control/sync traffic in IPC. The expectation for end-to-end IPC control is measured in the low 10's of μ s for the E610 and is expected to drop to a single digit μ s later. Some elements in multimedia traffic also bear similar requirements. Although some of the end-to-end delays can be quite long, the individual contribution of the arbitration in each node must be kept to a minimal. End-to-end budgets do not comprehend large delays within transmission nodes.
- **No-drop behavior and network congestion management** — The end node must be able to guarantee no-drop behavior for some TCs or some packets within TCs. As a termination point in receive, it is the end node's responsibility to properly control traffic coming from the network to achieve this end. For traffic generation in transmit, the end station must be able to positively respond to flow control from the network as the must have tool to prevent packet drop. It also needs to participate in network congestion management.
- **Compatibility with existing systems** — The DCB implementation needs to be usable by IT using known configurations and parameters, unless new ones are made expressly available. For example, DCB implementation cannot assume new knowledge regarding bandwidth allocation of traffic types that do not have known bandwidth requirements.

The layer 2 features of DCB implemented in the E610 are:

- **Multi-class priority arbitration and scheduling** — The E610 implements an arbitration mechanism on its transmit data path. The arbitration mechanism allocates bandwidth between TC in BWGs and between Virtual Machines (VMs) or Virtual Functions (VFs) in a virtualization environment. The BWGs can be used to control bandwidth and priority allocated to traffic types. Typically BWGs should be used to represent traffic types. TC arbitration allows control of bandwidth and priority control within BWGs as well as within the entire link bandwidth. The arbitration is designed to respect the bandwidth allocations to BWGs. The priority allocation allows minimization of delay for specific TCs. In the E610, TCs and user priorities are processed on a packet-by-packet basis based on the 802.1p identifier in the 802.1Q-tag.
- **Class-based flow control (PFC — Priority Flow Control)** — Class-based flow control functionality is similar to the IEEE802.3X link flow control. It is applied separately to the different TCs.
 - Transmit response to class-based flow control from the ingress switch it is connected to.
 - Receive class-based flow control commands to the switch in response to packet buffers filling status.
- **DMA queuing per traffic type** — Implementation of the DCB transmit, minimization of software processing and delays require implementation of separate DMA queues for the different traffic types. The E610 implements 128 descriptor queue in transmit and 128 descriptor queues in receive.
- **Multiple Buffers** — The E610 implements separate transmit and receive packet buffers per TC.
- **Rate-limiter per Tx queue** — Limiting the transmit data rate for each Tx queue.

Latency requirements:

Quantitative latency requirements are defined for a single 64-byte packet at the highest priority traffic class. Latency is defined separately for transmit and receive:

- **Transmit latency** — Measured from a tail update until the packet is transmitted on the wire. It is assumed that a single packet is submitted for this TC and its latency is then measured in the presence of traffic belonging to other TCs.
- **Receive latency** — Measured from packet reception from the wire and until the descriptor is updated on PCIe.

Figure 7-30 shows the latency requirements as previously defined.

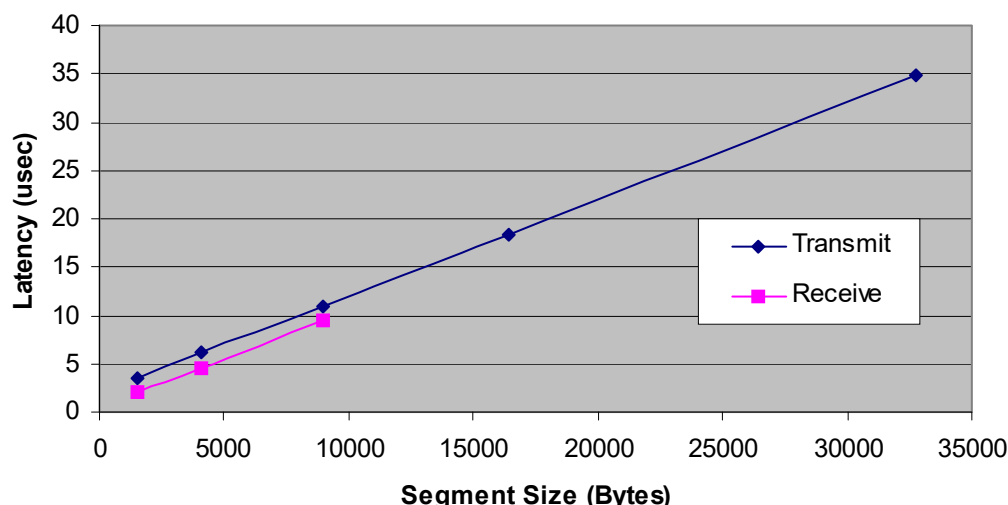


Figure 7-30. Latency Requirements

Note: In DCB mode, it is assumed all traffic is tagged (contains a VLAN header) — except for Layer2 frames with special Ethernet MAC Addresses that goes un-tagged. GMRP frames (special Ethernet MAC Addresses starting with 0x0180C20000) must, however, go tagged. Un-tagged packets must be delivered to the host and are assumed to belong to User Priority 0.

7.6.2 Transmit-Side Capabilities

When configured for DCB mode or when using the Tx rate-limiting functionality, the E610 software driver should only use advanced transmit descriptors. Refer to [Section 7.2.3.2.3](#).

7.6.2.1 Transmit Rate Scheduler (RS)

7.6.2.1.1 Basic Rate Control Operation

Rate control is defined in terms of maximum payload rate, and not in terms of maximum packet rate. This means that each time a rate controlled packet is sent, the next time a new packet can be sent out of the same rate controlled queue is relative to the packet size of the last packet sent. The minimum spacing in time between two starts of packets sent from the same rate controlled queue is recalculated in hardware on every packet again, by using the following formula:

$$\text{MIFS} = \text{PL} \times \text{RF}$$

Where:

- Packet Length (PL), is the Layer2 length (such as without preamble and IPG) in bytes of the previous packet sent out of that rate controller. It is an integer ranging from 64 to 9 K (at least 14-bits).

- RF = 10 Gb/s / target rate (rate factor) is the ratio between the nominal link rate and the target maximum rate to achieve for that rate-controlled queue. It is dynamically updated by software via the RTTQCNRC register or periodically according to the self-recovery algorithm run in software. It is a decimal number ranging from 1 to 1,000 (10 Mb/s minimum target rate). For example, at least 10 bits before the hexadecimal point and 14 bits after as required for the maximum packet length by which it is multiplied. For links at 1 Gb/s, the rate factor must be configured relatively to the link speed, replacing 10 Gb/s by 1 Gb/s in the above formula.
- Minimum Inter Frame Space (MIFS) is the minimum delay in bytes units, between the starting of two Ethernet frames issued from the same rate-controlled queue. It is an integer ranging from 76 to 9,216,012 (at least 24 bits). In spite of the 8-byte resolution provided at the internal data path, the byte-level resolution is required here to maintain an acceptable rate resolution (at 1% level) for the small packets case and high rates.

Note: It might be that a pipeline implementation causes the MIFS calculated on a transmitted packet to be enforced only on the subsequent transmitted packet.

Time Stamps — A rate-scheduling table contains the accumulated interval MIFS, for each rate-controlled descriptor queue separately, and is stored as an absolute Time Stamp (TS) relative to an internal free running timer. The TS value points to the time in the future at which a next data read request can be sent for that queue. Whenever updating a Timestamp:

$$\text{TimeStamp}(\text{new}) = \text{TimeStamp}(\text{old}) + \text{MIFS}$$

When a descriptor queue starts to be rate controlled, the first interval MIFS value is equal to 0 (TS equal to the current timer value) — without taking into account the last packet sent prior to rate control. When the TS value stored becomes equal to or smaller than the current free running timer value, it means that the switch is on and that the queue starts accumulating compensation times from the past (referred as a negative TS). When the TS value stored is strictly greater than the current free running timer value, it means that the switch is off (referred as a positive TS).

(CurrentTime) < TimeStamp <--> switch is off
(CurrentTime) >= TimeStamp <--> switch is on

MMW — The ability to accumulate negative compensation times that saturates to a Max Memory Window (MMW) time backward. MMW size is configured per TC via the *MMW_SIZE* field of the RTTQCNRM register, and is expressed in 1 KB units of payload, ranging from 0 up to 2 KB units (at least 11 bits). The *MMW_SIZE* configured in KB units of payload has to be converted in time interval *MMW_TIME* expressed in KB, before a new time stamp is checked for saturation. It is computed for each queue according to its associated Rate Factor (RF) using the following formula:

$$\text{MMW\_TIME} = \text{MMW\_SIZE} \times \text{RF}$$

Note: *MMW_TIME* is rounded by default to a 1 KB precision level and must be at least 31 bits long. Hence, the time stamp byte-level values stored must be at least 32 bits long for properly handling the wrap-around case. 29 bits are required for the internal free running timer clocked once every 8 bytes.

Whenever updating a time stamp verify:

$$\text{TimeStamp}(\text{old}) + \text{MIFS} \geq (\text{CurrentTime}) - \text{MMW\_TIME}$$

and then the time stamp is updated according to the non-saturated formula:

$$\text{TimeStamp}(\text{new}) = \text{TimeStamp}(\text{old}) + \text{MIFS}$$

Otherwise, enforced saturation by assigning:

$$\text{TimeStamp}(\text{new}) = (\text{CurrentTime}) - \text{MMW\_TIME} + \text{MIFS}$$

Note: Non-null MMW introduces some flexibility in the way controlled rates are enforced. It is required to avoid overall throughput losses and unfairness caused by rate-controlled packets over-delayed, consequently to packets inserted in between. Between two rate-limited packets spaced by at least the MIFS interval, non-rate-limited packets, or rate-limited packets from other rate-controlled queues, might be inserted. If a rate controlled packet has been delayed by more time than it was required for rate control (because of arbitration between VMs or TCs), the next MIFS accumulates from the last time the queue was switched on by the QCN rate scheduling table — and not from the current time. Refer to [Figure 7-31](#) for visualizing the effect of MMW.

MMW_SIZE set to 0b must be supported as well.

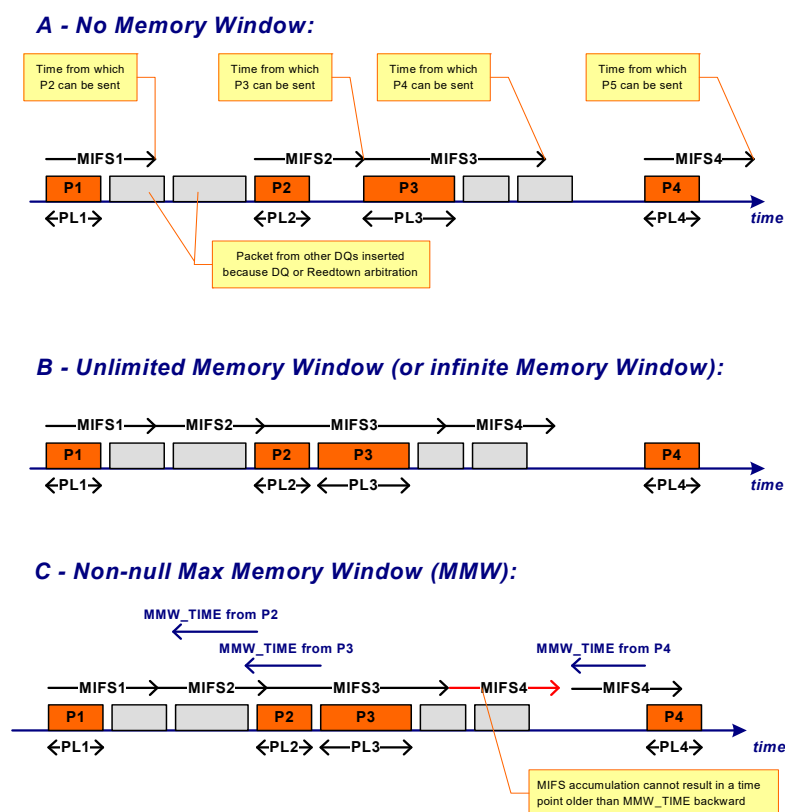


Figure 7-31. Minimum Inter-Frame Spacing for Rate-Controlled Frames (in Orange)

7.6.2.2 User Priority to Traffic Class Mapping

DCB-enabled software is responsible for classifying any Tx packet into one of the eight 802.1p user priorities, and to assure it is tagged accordingly by either software or hardware. The driver dispatches classified Tx traffic into the Tx queues attached to the proper TC, according to a UP-to-TC Tx mapping policy decided by the IT manager.

Caution: When translating XON/XOFF priority flow control commands defined per UP into commands to the Tx packet buffers, the E610 is required to use the same UP-to-TC Tx mapping table that software is using. The RTTUP2TC register must be configured by software accordingly. Refer to [Section 3.6.5.1.3](#) for details on priority flow control.

7.6.2.3 VM-Weighted Round-Robin Arbiters

The E610 implements VM-weighted arbiter(s) for virtualized environments and according to the following case:

- If DCB is enabled, there is one such arbiter per TC, arbitrating between the descriptor queues attached to the TC (one queue per VF). Bandwidth allocation to VMs is enforced at the descriptor plane, per each TC separately. The VM arbiter instantiated for each TC is aimed to elect the next queue for which a data read request is sent in case the TC is elected for transmission by the next level arbiter. For example, the TC weighted strict priority descriptor plane arbiter.
- If DCB is disabled, there is one single VM weighted arbiter, arbitrating between pools of descriptor queues, where a pool is formed by the queues attached to the same VF. Bandwidth allocation to VMs is enforced at the descriptor plane, between the pools, where queues within a pool are served on a frame-by-frame round-robin manner.

Refer to the different arbitration schemes where virtualization is enabled, as described in [Section 7.8](#).

Note: In this section, VM is considered a generic term used to refer to the arbitrated entity, whether it is a Tx descriptor queue within the TC or whether it is a pool of Tx descriptor queues. In the later case, pool parameters are allocated only to the lowest indexed queue within the pool, taken as the representation of the entire pool.

7.6.2.3.1 Definition and Description of Parameters

Credits: Credits regulate the bandwidth allocated to VMs. As part of the Weighted Round Robin (WRR) algorithm, each VM has pre-allocated credits. They are decremented upon each transmission and replenished cyclically. The ratio between the credits of the VMs represents the relative bandwidth percentage allocated to each VM (within the TC for the DCB enabled case). The E610 effectively maintains one table that represents these ratios. Note that credits can get negative values down to the maximum sized frame allowed on the TC/pool.

Note: The absolute value of the credits has no direct bearing on the allocation of bandwidth. The ratio between the credits does. However, since credits can accumulate only up to twice the credit refills, the refills should be allocated as low as possible but must be set greater than the maximum sized frame allowed on the TC or on the pool.

WRR: The algorithm implemented in the E610 for VM arbitration.

Table 7-51 (T1) defines the VMs and their bandwidth allocation. The following elements are defined in this table:

Table 7-51. Bandwidth Allocation to VMs

T1: VM Bandwidth Allocation			
VM _N	VM Refill	VM Max Credits	VM Min Credits
0	MSS:1,024 KB	2xMSS:2,048 KB	-MSS
1	MSS:1,024 KB	2xMSS:2,048 KB	-MSS
2	MSS:1,024 KB	2xMSS:2,048 KB	-MSS
3	MSS:1,024 KB	2xMSS:2,048 KB	-MSS
...			
15	MSS:1,024 KB	2xMSS:2,048 KB	-MSS
Notes: 1. Due to a pipelined implementation, the VM credits range is enlarged by one MSS, beyond negative limits. 2. All values are implemented with 64-byte granularity (a value of one corresponds to 64 bytes of credit).			

VM: Configuration — The unique Tx descriptor queue attached to a VF within a TC, or the pool of Tx descriptor queues attached to the same VF.

VM Credit Refill: Configuration — The E610's WRR algorithm implement credit refill as the technique for percentage allocation to VMs. The credits refill are added to each VM credit count on each completion of the full round of the algorithm (after all the VMs had their chance to send what they had in store or at least one frame).

The E610's driver needs to calculate the VM credit refill to match the percent allocated through management (such as in the MIB). Since the WRR arbitration is self timed, the ratio between the credits refill is the only defining parameter for the VMs. However, the refills must be greater or equal to the maximum sized frame allowed on the TC or on the pool to guaranty transmission of at least one frame on each recycling round. The E610 allows a value of 1.5 KB to 1,024 KB for a dynamic range of x1000.

VM Maximum Accumulated Credits: Deducted from Refill Configuration — To prevent the use of stale credits, the number of credits each VM can accumulate upon refill is limited. The credits for each VM can only reach twice their refill. The maximum range for the credits is thus -9.5 KB to 2,048 KB, assuming negative credits can accumulate up to a maximum sized frame (9.5 KB if jumbo frames are allowed), and where positive credits can accumulate up to twice the maximum credit refill.

VM Credits: Run time parameter — VM credits is a running counter for each VM. It holds the number of available credits in 64-byte units. The algorithm runs sequentially between the VMs and enables transmission for those VMs that have enough pending credits (their credit number is greater than zero).

Table 7-52. Registers Allocation for Tx VM Arbiters

Attribute	Tx VM Arbiter
VM Control registers	RTTDT1C
VMC Status registers	RTTDT1S
VM credit refill	CRQ
VM credits	CCC

7.6.2.3.2 WRR Arbiter Algorithm

Note: This feature should be combined with configuring at least twice the sum of $T1[VM]$. Refills over a TC to its corresponding $T2.[TC]MaxCredits$. See [Section 4.6.10.4.1](#) for details.

Round Robin — The round-robin aspect of the VM WRR arbiter resides in the fact that once a VM has been granted for a data read request, the next VMs are checked in a cyclic round-robin order, even if the granted VM still has credits for another data read request.

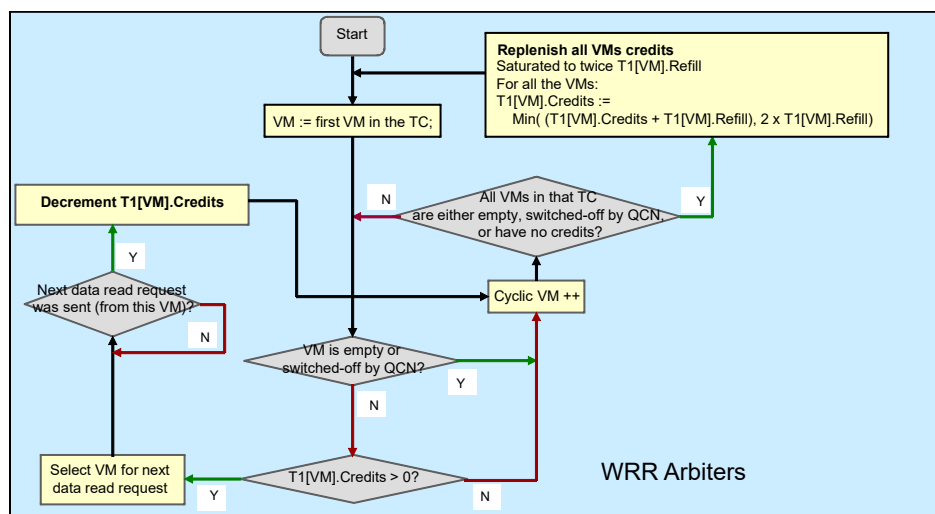


Figure 7-32. Tx VM WRR Arbiters Operation

7.6.2.4 Tx TC Weighted Strict Priority Arbiters

In DCB, multiple traffic types are essentially multiplexed over the Ethernet 10 Gb/s network. There is a need to allow different behavior to different traffic flows as they pass-through multiple Ethernet switches and links. For example, for LAN, SAN and IPC connections are consolidated on a single Ethernet link. Each traffic type (BWG) is guaranteed the bandwidth it has been allocated and is prevented from usurping bandwidth from other types. However, if a BWG does not use its bandwidth, that bandwidth is made available to the other BWGs. The same holds for TCs within a BWG. If allocated some bandwidth, TCs are guaranteed to have it, and if unused, that bandwidth can be used by the other TCs within the BWG. Information regarding bandwidth allocation for some TCs might not be available. In the case of LAN, the entire allocation of bandwidth within the LAN link is typically undefined in today's networks. The arbitration scheme includes Group Strict Priorities (GSP) to cover for that. TCs for which the GSP bit is set are limited by the total throughput allocated to their BWG rather than to TC allocation.

Link bandwidth is divided among the BWGs for guaranteed minimum behavior. For example: LAN: 4 Gb/s, SAN: 4 Gb/s, IPC: 2 Gb/s. The E610 supports two types of bandwidth allocation within BWGs. TCs can be either allocated bandwidth or be used as in strict priority. If a TC does not use all of its allocated bandwidth, that bandwidth is recycled to other TCs in the BWG.

The E610 implements two replications of the weighted TC arbiter:

- One in the descriptor plane, arbitrating between the different descriptor queues, deciding which queue is serviced next. It is aimed to enforce the TC bandwidth allocation and prioritization scheme for a case when PCIe bandwidth is smaller than the link bandwidth.

- A second in the packet plane, at the output of the packet buffers, deciding which packet to transmit next. It is aimed to enforce the TC bandwidth allocation and prioritization scheme for a case when PCIe bandwidth is greater than the link bandwidth.

The condition for entry into the bandwidth allocation algorithm sequence differs for the descriptor and data arbiters:

- The descriptor arbiter queries whether there is at least one queue attached to a TC that is not empty, not switched off by the rate scheduler, with positive VM weighted arbiter credits (when relevant), and for which the destined packet buffer has room for the worst case maximum sized frame. This last condition is controlled by `RTTDCS.BPBFSM`.
- The packet arbiter queries whether the packet buffer has a packet to send and whether it is not stalled by priority flow control.

7.6.2.4.1 Definition and Description of Parameters

User Priority (UP): There are eight traffic priorities, determined by 802.1p tag bits on an Ethernet link. The *Q-Tag* field holds UP's. Per 802.1p, Priority #7 is the highest priority. User priorities are assigned by the application or the system to certain usage classes, such as manageability, IPC control channel, VoIP. And additional bit within the VLAN TCI field (bit 5 - DEI) defines whether the packet has a no drop requirement. This bit is not being used by DCB mechanisms.

User Bandwidth Group (UBWG): A user bandwidth group is a management parameter that is a binding of user priorities into bandwidth groups for provisioning purposes. The hardware implementation does not recognize the UBWG entity.

Traffic Class (TC): Incoming packets are placed in traffic classes. Per the DCB functional specification, there might be a 1:1 mapping between UP and TC, or more than one priority can be grouped into a single class. Such grouping does not cross boundaries of traffic BWGs. The E610 implements eight or four TCs and maps them to UP's according to a programmable register. This provides the best flexibility for the IT manager. However, when more than one UP is mapped to the same TC, they must have the same no-drop policy network wide.

Packet Buffer (PB): TCs are mapped to packet buffers in a straightforward 1:1 mapping. Packets are also placed in packet buffers based on their class assignments.

Traffic Bandwidth Group (BWG): For bandwidth allocation and grouping, one or more TC can be grouped into a Traffic Bandwidth Group (BWG). A BWG is a logical association at a node, and has no markings inside a packet header. End stations and switches are independent in their definition and allocation of grouping of different TCs. Consistency of behavior throughout the network is handled by the UBWG provisioning mechanism.

One or more TCs can be grouped in a BWG. BWGs are allocated a percentage of bandwidth of available Ethernet link. The allocated bandwidth for BWG can be further divided among the TCs that are inside the BWG.

Credits: Credits regulate the bandwidth allocated to BWGs and TCs. As part of the WSP algorithm, each BWG and TC has pre-allocated credits. Those are decremented upon each transmission and replenished cyclically. The ratio between the credits of the BWGs represents the relative bandwidth percentage allocated to each BWG. The ratio between the credits of the TCs represents the relative bandwidth percentage allocated to each TC within a BWG. The E610 effectively maintains one table that represents both ratios at once. Note that credits can get negative values down to the maximum sized frame allowed on the TC.

Maximum Credit Value: The maximum credit value establishes a limit for the running sum of credits allotted to a class or group. This value prevents stacking up stale credits that can be added up over a relatively long period of time and then used by TCs all at once, altering fairness and latency.

Note: The absolute value of the credits has no direct bearing on the allocation of bandwidth. The ratio between the credits does. However, the absolute value might have substantial impact on the algorithm behavior. Larger absolute values can impact the latency of high priority queues and their ability to serve bursts with minimum latency, whereas too small credit values might impact the correct functionality in presence of jumbo frames. The speed of the algorithm implementation should also be taken into account. The value of the maximum credit limit are also in principle not part of the main WSP algorithm. However, they impact the fairness of bandwidth reallocation between queues in case some queues do not transmit the full amount they have been permitted to. Also, small values prevent correct functionality of jumbo frames. From high-level simulations it appears that credits should be allocated as low as possible based on the speed of the algorithm. Maximum credit values for TCs should be 1.5x to 2x the size of the maximum entity expected in that TC.

Weighted Strict Priority (WSP): The algorithm implemented in the E610 for TC arbitration.

Group Strict Priority (GSP): Refer to the sections that follow for details.

Link Strict Priority (LSP): Refer to the sections that follow for details.

Table 7-53 (T2) defines the TCs, their association to BWGs and their bandwidth allocation. The following elements are defined in this table:

Table 7-53. Bandwidth Allocation to TCs and BWGs

T2: Traffic Class Bandwidth Allocation Within a BWG						
TC _N	BWG	TC Refill	TC Max Credits	LSP	GSP	TC Min Credits (according to GSP 0/1)
0	1	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
1	2	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
2	0	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
3	1	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
4	2	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
5	3	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
6	1	64 bytes to 32 KB	64 bytes to 256 KB	0/1	1/1	-MSS / -2,048 KB
7	0	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB

Notes:

1. Due to a pipelined implementation, the TC credits range is enlarged by one MSS in both directions, beyond its positive and negative limits.
2. All values are implemented with 64-byte granularity (a value of one corresponds to 64 bytes of credit).

TC: Configuration — The traffic type associated to the packet buffer where incoming packets are kept before transmission (or discard — not implemented in transmit in the E610). TC7 is the highest priority TC.

BWG: Configuration — Traffic BWG that a TC belongs to.

TC Credit Refill: Configuration — The E610's WSP algorithm implements credit refill as the technique for traffic class percentage allocation. The credits refill are added to each TC credit count on each completion of the full round of the algorithm (after all TCs had their chance to send what they had in store and if they had credits for it).

The E610's driver needs to calculate the TC Credit refill to match the percentage allocated through management (in the MIB). The TC credit refill table includes, in one table both the TC and the BWG. Since the WSP arbitration is self timed, the ratio between the credits refill is the only defining parameter for the TC and BWG. The absolute values of the refill have significance as to the rate of distribution

between the queues and can have impact on latency and on the momentary stability of the bandwidth fairness. Results from simulations indicate that the quantum for the refill should be small to prevent large swings. It should not be too small as to create overhead in the mechanism due to the execution time; however. The E610 allows a value of 64 bytes to 32 KB, A dynamic range of x500. The usage model is likely to call for a smallest refill value of 256 bytes to 512 bytes. This leaves a dynamic range of 80x-200x. For example, if a queue is assigned 1% and this is translated into a 256-byte increment, another assigned with 99% would have a refill value of 25344 bytes.

TC MaxCredit: Configuration — To prevent use of stale credits, the number of credits each TC can accumulate upon refill is limited. The TC credit can only reach *MaxCredit*, beyond that its value gets recycled to other queues. Refer to the recycling mode for more details. The maximum range for the *MaxCredit* is 256 KB.

Note: Full testing of many values for maximum value is unnecessary. Hierarchical testing should be applied. For the random test, four to six values should be sufficient. It is important that the testing includes values that are relevant to the interesting zone of maximum value. For example, in the 0.8x to 2x the relevant largest entity in the class. Although class with an expected shorter packet could use a smaller *MaxCredit*, it is recommended that testing fully covers the cases where all the classes have similar values of *MaxCredit*, as it is a possible variant use of the algorithm.

Link Strict Priority (LSP): Configuration — If set, this bit specifies that this TC can transmit without any restriction of credits. This effectively means that this TC can take up the entire link bandwidth, unless preempted by higher priority traffic. If this bit is set, *TC.CreditRefill* must be set to 0b to ensure fair bandwidth allocation. Preferably, the algorithm implementation should disregard non-zero values in all its calculations.

Group Strict Priority (GSP): Configuration — This bit defines whether strict priority is enabled or disabled for this TC within its BWG. If *TC.GSP* is set to 1b, the TC is scheduled for transmission using strict priority. It does not check for availability of TC.Credits. It does check whether the BWG of this TC has credits (such as the amount of traffic generated from this TC is still limited by the BWG allocated for the BWG (T3. BWGP). If this bit is set, *TC.CreditRefill* values can be set to 0b, if a non-zero value is configured, TC credits are reduced first from the GSP TC and if reached to zero from other TCs in the group, if the refill credits are configured to zero the TC credits are reduced from the other TCs in the BWG.

Note: Since the *TC.GSP* parameter relates to individual TCs, some BWGs might have both TCs with bandwidth allocation and TCs with GSP. This is a hybrid usage mode that is complex to validate and is possibly secondary in importance.

Usage Note — It is possible that a TC using LSP dominates the link bandwidth if there are no packets waiting and eligible in higher priority TCs. To guarantee correct bandwidth allocation, all TCs with the unlimited bit set should be in the same traffic BWG (high priority group). Note that this is different than a typical DCB deployment considered where BWG is created with functional grouping, like LAN, SAN, and IPC etc. TCs with the *LSP* bit set should be the first to be considered by the scheduler (the first TCs). For example, from queue 7 to queue 5 with the other five TCs for groups with bandwidth allocation. These are not strict requirements, if these rules are not followed, undesirable behavior could occur in some cases. A group containing only TCs with the unlimited bit set effectively has zero BWG credits since TCs with the LSP unlimited bit set should have TC credits set to zero. Use of LSP / TC.GSP should be restricted to UPs/TCs that service trusted applications.

TC Credits: Run-time parameter — TC credits is a running algebraic counter for each TC. It holds the number of available credits in 64-byte units. The algorithm runs sequentially between the TCs and enables transmission for those TCs that have positive pending credits. Note that credits can get negative values down to the maximum sized frame allowed on the TC.

Table 7-54 (T3) defines the hierarchy of BWG similarly to T2 defining the hierarchy within the BWG's. T3 implementation should include eight rows.

Table 7-54. Line Bandwidth Allocations to Bandwidth Groups (BWG) (with example)

T3: Line bandwidth allocation to Traffic Bandwidth Groups (BWG)				
BWG	BWG Refill Credits	BWG MaxCredit	BWG Credit	Description
0	= $\Sigma[TC]$ Credit Refill	= $\Sigma[TC]$. MaxCredit	-/+ 2,048 KB	IPC
1	= $\Sigma[TC]$ Credit Refill	= $\Sigma[TC]$. MaxCredit	-/+ 2,048 KB	SAN
2	= $\Sigma[TC]$ Credit Refill	= $\Sigma[TC]$. MaxCredit	-/+ 2,048 KB	LAN
3	= $\Sigma[TC]$ Credit Refill	= $\Sigma[TC]$. MaxCredit	-/+ 2,048 KB	Manageability
-				
-				
-				
7				

Note: Due to a pipelined implementation, the BWG credits range is enlarged by one MSS in both directions, beyond its positive and negative limits.

BWG: Configuration — This is the number of the traffic BWG that is three bits wide. This field corresponds to the *TC.BWG* field in [Table 7-53](#).

BWG Refill Credits: A virtual number — The credits provisioned for this BWG. The credits ratio between the BWG's should reflect the ratio of bandwidth between the BWG's. In the actual implementation, this number is the sum of the credit Refills of the TC's associated with this BWG.

BWG MaxCredit: A virtual number — The maximum credits for a BWG. Credits in the BWG.Credit counter are limited to this value. Credits that should have been refilled above this value are lost. In effect, due to the self-timed cyclic nature of the WSP algorithm, those credits are distributed between all BWG's. In the actual implementation, this number is the sum of the MaxCredit of the TC's associated with this BWG.

BWG.Credit: Run-time parameter — A running algebraic counter that is decremented for each transmission. At the end of each cycle, this counter is synchronized with the sum of the TC.Credit counter associated with this BWG. The synchronization algorithm depends on the recycling mode. refer to the sections that follow for details about arbitration configurations. Note that credits can get negative values down to the maximum sized frame allowed on the BWG.

7.6.2.4.2 Arbiters Conventions

The WSP scheme previously described is written with the data plane arbiter in mind. However, the same scheme is used by the transmit descriptor plane arbiter and a subset of it is used by the receive data arbiter. To distinguish between the two arbiters, attributes of the each arbiter are prefixed as depicted in [Table 7-55](#).

Table 7-55. Attributes of Tx Arbiters

Attribute	Tx Packet Arbiter	Tx Descriptor Arbiter
TC	P-TC	D-TC
BWG	P-BWG	D-BWG
TC Credit Refill	P-TC Credit Refill	D-TC Credit Refill
TC MaxCredit	P-TC MaxCredit	D-TC MaxCredit
LSP	P-LSP	D-LSP

Table 7-55. Attributes of Tx Arbiters [continued]

Attribute	Tx Packet Arbiter	Tx Descriptor Arbiter
GSP	P-GSP	D-GSP
TC Credits	P-TC Credits	D-TC Credits
BWG Refill Credits	P-BWG Refill Credits	D-BWG Refill Credits
BWG MaxCredits	P-BWG MaxCredits	D-BWG MaxCredits
BWG Credits	P-BWG Credits	D-BWG Credits

Table 7-56 lists the register fields that contain the relevant attributes from Table 7-55.

Table 7-56. Registers Allocation for Tx TC Arbiters

Attribute	Tx Packet Arbiter	Tx Descriptor Arbiter
TC Control registers	RTTPT2C	RTTDT2C
BWG	BWG	BWG
TC credit refill	CRQ	CRQ
TC MaxCredit	MCL	MCL
LSP	LSP	LSP
GSP	GSP	GSP
TC credits	CCC	CCC

7.6.2.4.3 Tx TC WSP Arbitration Configurations

There are two parameters influencing the Tx TC WSP arbitration:

- RR/WSP
- Recycle mode

A separate set of configuration parameters exists for each of the three TC arbiters, as listed in Table 7-57.

Table 7-57. Configuration Parameters for the Tx and Rx TC Arbiters

Parameter	Tx Packet Arbiter	Tx Descriptor Arbiter	Rx Packet Arbiter
	RTTPCS	RTTDCS	RTRPCS
RR/WSP Mode	TPPAC	TDPAC	RAC
Recycle Mode	TPRM	TDRM	RRM

The impact of these settings on the arbitration is described in the sections that follow.

7.6.2.4.3.1 RR/WSP

When this bit is set, the arbitration is in WSP mode. When reset, the arbitration is in flat frame-based round robin mode. In RR mode, one frame is transmitted from each packet buffer in its turn. BWG and TC parameters do not apply.

7.6.2.4.3.2 Recycle Mode

Architecture Overview of Recycle — As a result of GSP transmits and TCs that reach their maximum credit limit, the credit count of a BWG might not match the total credit count of its TCs (refer to the sections that follow for more details). It is not merely an arithmetic issue. The WSP algorithm, dual hierarchy behaves as a maximum allocation algorithm within the BWG's and minimum allocation algorithm between the BWG's. Since the recycle is self timed, when a BWG does not transmit all of its allocated bandwidth within a cycle, at the end of the cycle its bandwidth is in effect reallocated to all BWG's. This results in a minimum allocation behavior. Inside the BWG's; however, this notion of self timing does not exist. Some explicit mechanism is required to recycle bandwidth within a BWG rather than to all the BWG's — The requirement to have a minimum allocation behavior.

- A BWG credit count might not match the total credit count of its TCs in the following cases:
 - **A TC is defined as GSP** — When a GSP is selected, the BWG credits are decremented but no TC is deducted. Therefore, the BWG credit count would be lower than the credit count of its TCs.
 - **Max credits during refill** — If a TC reaches its max credits value during refill, some credits are lost for that TC. However, the BWG for that TC is provided with the full refill count. Therefore, the BWG credit count would be higher than the credit count of its TCs.
- One bit per TC arbiter governs the recycle mode of the WSP algorithm:
 - **0: No Recycling** — At the end of each full arbitration cycle all TCs are refilled with their TC.Refill up to their TC.MaxCredits values. All BWG.Credit are loaded by the sum of the BWG TC.Credit.
 - **1: Recycle** — TC credits for TCs that have reached their maximum are recycled to other TC' of the BWG. The operation is calculated based on the BWG.Credit and the TC.Credits after their refill. The difference between them is the BWG.Recycle value.
 - **Positive BWG.Recycle** — The recycle algorithm adds credits from the BWG.Recycle to the TC.Credits starting from the highest priority TC in the BWG down, considering the TC.MaxCredit, until BWG.Recycle is zero.
 - **Negative BWG.Recycle** — The recycle algorithm subtracts credits from the BWG.Recycle to the TC.Credits starting from the lowest priority TC in the BWG up, until BWG.Recycle is zero.

7.6.2.4.4 Tx TC WSP Arbiter Algorithm

The Transmit Packet Plane Arbitration Control (TPPAC) bit in the RTTPCS registers determines the scheduler type (RR or WSP).

Strict Priority — The strict-priority aspect of the TC WSP arbiter resides in the fact that once a TC has been granted for a data read request or for transmission, the highest priority TCs are checked (again) in a strict-priority order, starting from TC7, even if the granted TC still has credits for another data read request or transmission.

Note: The descriptor plane arbiter cannot issue a data read request unless there is an unused request for data. Therefore the arbiter stalls in its current state each time there are no data read requests available. The next arbitration decision is only done once there is at least one free data read request.

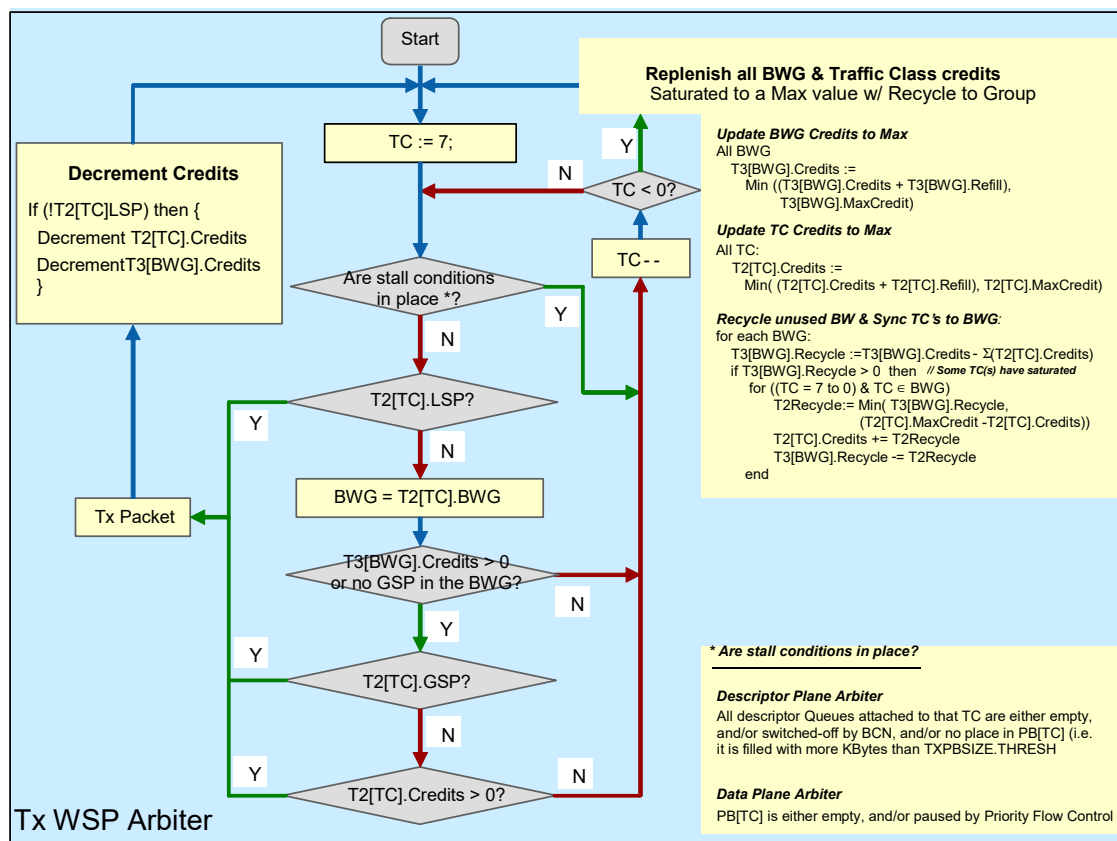


Figure 7-33. Tx TC WSP Arbiters Operation

7.6.3 Receive-Side Capabilities

7.6.3.1 User Priority to Traffic Class Mapping

To enable different TC support for incoming packets and proper behavior per TC, the E610's receive packet buffer is segmented into several packet buffers. The E610 supports the following configurations of packet buffer segmentation:

- DCB disabled — Single buffer of 384 KB
- DCB enabled with 4 TCs — 4 buffers, 96 KB each
- DCB enabled with 8 TCs — 8 buffers, 48 KB each
- DCB enabled with 8 TCs — 8 buffers, buffers 3:0 are 72 KB and buffers 7:4 are 24 KB

Incoming packets are transferred from the packet buffers into data buffers in system memory. Data buffers are arranged around descriptor rings described in [Section 7.1.5.5](#). Each descriptor queue is assigned dynamically to a given TC (and therefore to a packet buffer) as described in [Section 7.1.3](#).

Configuration registers:

- The size of each buffer is defined by the RXPBSIZE[0-7] registers. Note that it is possible to configure the buffers at 1 KB granularity.
- A received packet is assigned by the E610 to a TC, and is thus routed to the corresponding Rx packet buffer according to its *User Priority* field in the 802.1Q tag and according to a UP to TC mapping table loaded into the RTRUP2TC register.

Caution: Different UP to TC mappings can be loaded in each direction Tx and Rx, as per RTTUP2TC and RTRUP2TC registers, respectively. But in such a case, when a packet is looped back by the internal VM to VM switch, it is routed to the Rx packet buffer that corresponds to the TC that was used in Tx.

7.6.3.2 Rx PB Weighted Strict Priority Arbiter

The E610's Rx arbiter determines the order in which packets are written from the different packet buffers into system memory. Note that each packet buffer by itself is drained in the order packets arrived, as long as it deals with packets destined to the same Rx queue.

The arbitration algorithm between the receive packet buffers is WSP, similar to the TC scheme on the transmit side. Motivation for this scheme is as follows:

1. The major consideration is to prevent any delay in delivery of high-priority traffic.
2. Allocation of credits controls the bandwidth allocated to the different packet buffers.
3. A secondary mean of bandwidth allocation is the priority flow control. By altering the flow control high watermark, the E610 can effectively (if coarsely) regulate bandwidth allocation to types of traffic.

Table 7-58 (T4) defines the TCs and their bandwidth allocation. The following elements are defined in this table:

Table 7-58. Bandwidth Allocation to Traffic Classes and Bandwidth Groups

T4: Packet Buffer Bandwidth Allocation Within a BWG						
PB	BWG	PB Refill	PB Max Credits	LSP	GSP	PB Min Credits (according to GSP 0/1)
0	1	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
1	2	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
2	0	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
3	1	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
4	2	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
5	3	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
6	1	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
7	0	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB

Note:

1. Due to a pipelined implementation, the PB credits range is enlarged by one MSS in both directions, beyond its positive and negative limits.
2. All values are implemented with 64-byte granularity (a value of one corresponds to 64 bytes of credit).

Table 7-59 (T5) defines the hierarchy of BWG similarly to T4 defining the hierarchy within the BWG's. T4 implementation should include eight rows.

Table 7-59. Packet Buffer Allocations to BWGs (with Example)

T5: Line Bandwidth Allocation to Traffic BWGs				
BWG	BWG Credit Refill	BWG MaxCredit	BWG Credit	Description
0	= $\Sigma[\text{TC}]$ Credit Refill	= $\Sigma[\text{TC}]$. MaxCredit	-/+ 2,048 KB	IPC
1	= $\Sigma[\text{TC}]$ Credit Refill	= $\Sigma[\text{TC}]$. MaxCredit	-/+ 2,048 KB	SAN
2	= $\Sigma[\text{TC}]$ Credit Refill	= $\Sigma[\text{TC}]$. MaxCredit	-/+ 2,048 KB	LAN
3	= $\Sigma[\text{TC}]$ Credit Refill	= $\Sigma[\text{TC}]$. MaxCredit	-/+ 2,048 KB	MGMT
-				
-				
-				
7				

The attributes of the Rx packet arbiter are described in [Section 7.6.2.4.2](#).

7.6.3.2.1 Rx TC Arbitration Configurations

Table 7-60 lists the register fields that control the Rx arbiter.

Table 7-60. Registers Allocation for DCB Rx Arbiters

Attribute	Rx Packet Arbiter
PB Control registers	RTRPT4C
PB Status registers	RTRPT4S
BWG	BWG
PB credit refill	CRQ
PB MaxCredit	MCL
LSP	LSP
GSP	GSP
PB credits	CCC

Configuration parameters for the Rx packet arbiter are defined and listed in [Section 7.6.2.4.1](#).

7.6.3.2.2 Rx TC WSP Arbiter Algorithm

The Rx packet arbiter operates in RR or WSP mode, configured through the *RAC* bit in the RTRPCS register. The WSP arbiter operation is described as follows.

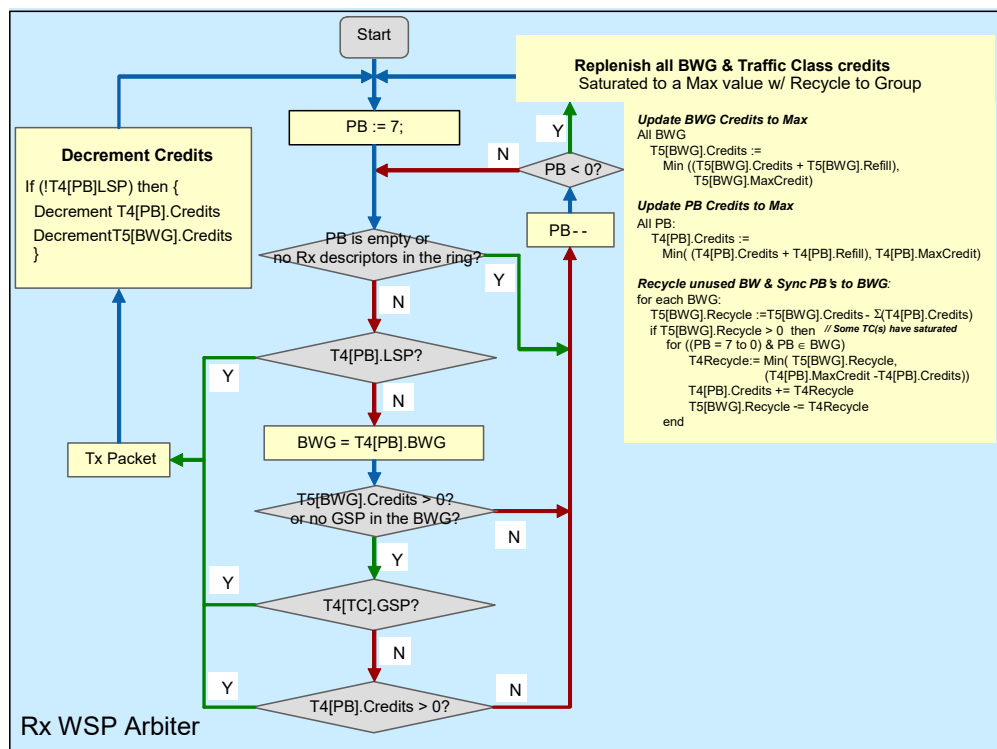


Figure 7-34. Rx Packet WSP Arbiter Operation

7.7 Precise Timing Protocol (PTP)

7.7.1 Overview

IEEE 1588 addresses the clock synchronization requirements of measurement and control systems. The protocol supports system-wide synchronization accuracy in the sub-microsecond range with minimal network and local clock computing resources. The protocol is spatially localized and allows simple systems to be installed and operate.

The IEEE802.1AS standard specifies the protocol used to ensure that synchronization requirements are met for time sensitive applications, such as audio and video or industrial usage, across Bridged and Virtual Bridged Local Area Networks consisting of LAN media where the transmission delays are almost fixed and symmetrical; for example, IEEE 802.3 full duplex links. This includes the maintenance of synchronized time during normal operation and following addition, removal, or failure of network components and network re-configuration. It specifies the use of IEEE 1588 specifications where applicable.

Note: The PTP mechanism activation is possible in full-duplex mode only.

Note: The clock used for PTP is a 312.5 MHz clock, so the limit of the accuracy of all PTP operations is 3.2 ns.

7.7.2 Flow and Hardware/Software Responsibilities

The operation of a Precision Time Protocol (PTP) enabled network is divided into two stages: initialization and time synchronization.

At the initialization stage, every master-enabled node starts by sending sync packets that include the clock parameters of its clock. Upon receipt of a sync packet, a node compares the received clock parameters to its own and if the received parameters are better, then this node moves to a slave state and stops sending sync packets. While in slave state, the node continuously compares the incoming packet to its currently chosen master and if the new clock parameters are better, than the master selection is transferred to this master clock. Eventually the best master clock is chosen. Every node has a defined time-out interval that if no sync packet was received from its chosen master clock it moves back to a master state and starts sending sync packets until a new best master clock (PTP) is chosen. The protocol supports also a mechanism by which the master (and possibly a backup master as well) are chosen upfront by the network manager.

The time synchronization stage is slightly different between the flow described in the 1588 standard and the 802.1AS.

7.7.2.1 1588 Time Synchronization Flow

The master sends periodically sync packets that their transmission time is sampled by the hardware (as close as possible to the MDI pins). This time is indicated as '**t1**'. This sample time is appended to the Sync packet ('1 step') or transmitted in a follow on packet ('2 step'). E610 supports only the '2 step' option. The slaves samples its reception time (as close as possible to the MDI pins). This time is indicated as '**t2**'. It then respond by a Delay_Req packet for each Sync packet. The slave and the master sample its transmission time and reception time respectively. This time values are indicated as '**t3**' and '**t4**' respectively. Then the master replies with a Delay_Resp packet that carries the sampled reception time of the Delay_Req packet. Following the reception of the Delay_Resp packet, the slave has the knowledge of all 4 sampled times: t1, t2, t3 and t4 as illustrated in Figure 7-35. Using these values, the slave calculates the time delta relative to the master and uses it to adjust its own timer.

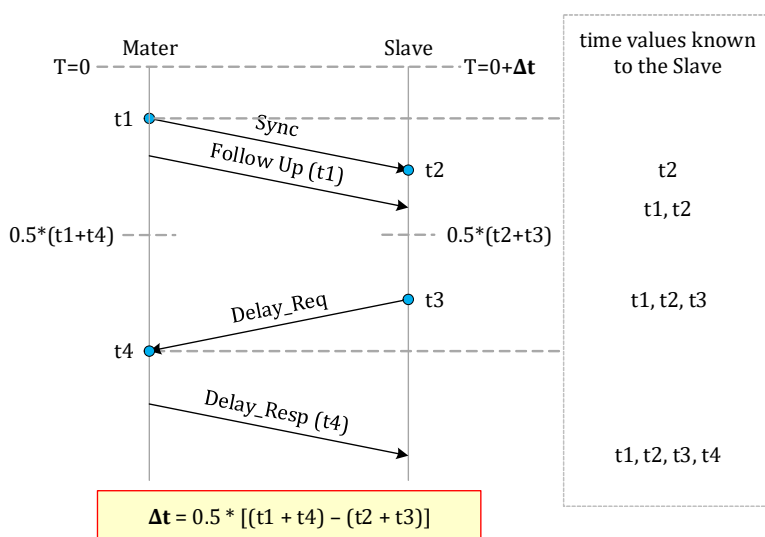


Figure 7-35. 1588 Sync Flow for Offset Calculation

7.7.2.2 802.1AS Time Synchronization Flow

Link Delay Calculation (illustrated in Figure 7-36): This flow is common to 802.1AS and 1588 standards. Each node (master and slave) is responsible to calculate the link delay between itself and the other node. Specifically, the slave calculates the link delay between itself and the master. The slave sends periodically PDelay_Req packets and sample its transmission time. This time is indicated as '**t1**'. The master samples the reception time of these packets, indicated as '**t2**'. The master replies with PDelay_Resp packets and sample its transmission time. This time is indicated as '**t3**'. The slave sample its reception time, indicated as '**t4**'. The sampled transmission time is transmitted to the slave in a Follow up packet. Using these values the slave calculates the link delay. indicated as '**td**'. It is critical to sample these PDelay packets by a monotonic timer that is not adjusted minimizing link delay measurement errors.

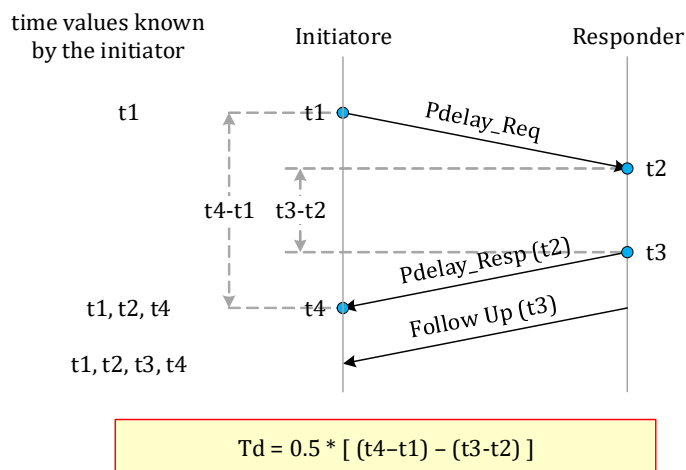


Figure 7-36. 802.1AS and 1588 Link Delay Measurement Flow

Time Difference Calculation (illustrated in Figure 7-37): The master sends periodically sync packets that their transmission time is sampled by the hardware. This time is indicated as 't1'. This sample time is transmitted in a follow on packet ('2 step' option). The slaves samples its reception time, indicated as 't2'. Using these values and the knowledge of the link delay, the slave calculates the time delta relative to the master and uses it to adjust its own timer.

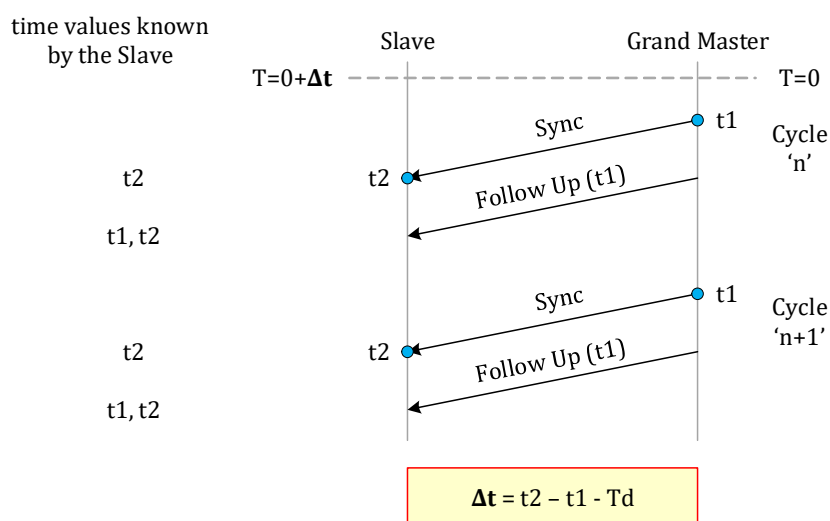


Figure 7-37. 802.1AS and 1588 Link Delay Measurement Flow

7.7.2.3 PTP Indications in Rx and Tx Packet Descriptors

Some indications need to be transferred between software and hardware regarding PTP packets. On the Tx path, software should set the 1588 bit in the Tx packet descriptor (bit 9). On the Rx path, hardware has two indications to transfer to software, one is to indicate that this packet is a PTP packet (whether time stamp is taken or not). This is also for other types of PTP packets needed for management of the protocol and this bit is set only for the L2 type of packets (the PTP packet is identified according to its EtherType). PTP packets have the *L2 Packet* bit in the packet type field set (bit 11 in the receive descriptor) and the EtherType matches the filter number set by software in the ETQF registers to filter PTP packets. The UDP type of PTP packets do not need such indication since the port number (319 for event and 320 for all the rest PTP packets) directs the packets toward the PTP application. The second indication is *RDESC.STATUS.TS* (bit 16) to indicate to software that time stamp was taken for this packet. Software needs to access the time stamp registers to get the time stamp values.

7.7.3 Hardware PTP Elements

All PTP hardware elements are reset to their initial values (as defined in [Section 8.2.2.19](#)) upon Master reset.

7.7.3.1 System Time Structure and Mode of Operation

The *SYSTIME* is a 96 bit register is composed of: *SYSTIMR*, *SYSTIMEL* and *SYSTIMEH* registers: The *SYSTIMR* register holds the sub-nanosecond fraction, the *SYSTIMEL* register holds the nanosecond fraction and the *SYSTIMEH* register holds the sec fraction of the time (note that the upper two bits of the *SYSTIMEL* register are always zero while the max value of this register is 999,999,999 decimal).

- **Initial Setting** — Setting the initial time is done by direct write access to the *SYSTIME* register. Software should set first the *SYSTIMEL* and then set the *SYSTIMEH*. Setting the *SYSTIMR* is meaningless while it represents sub-nanosecond units.
- **Run Time** — During run time the *SYSTIME* timer value in the *SYSTIMEH*, *SYSTIMEL* and *SYSTIMR* registers, is updated periodically each PTP clock (of 3.2 ns) according to the following formula:
 - Define: $INC\_TIME == 3.2\text{ ns} +/- TIMEINCA.Incvalue * 2^{-32}\text{ ns}$. Add or subtract the *TIMEINCA.INCVALUE* is defined by *TIMEINCA.ISGN* (where 0b means Add and 1b means Subtract)
 - Then: $SYSTIME = SYSTIME + INC\_TIME$
- **Reading the *SYSTIME*** register by software is done by the following sequence:
 - Read the *SYSTIMEL* register.
 - Read the *SYSTIMEH* register.
- **Dynamic update of *SYSTIME*** registers is done by using the *TIMADJ* registers by the following flow:
 - Write the *TADJL* value and its *SIGN* to the *TIMADJ* register (the *SIGN* bit indicates if the *TADJL* value should be added or subtracted).
 - Following the write access to the *TIMADJ* register, the hardware repeats the following two steps at each PTP clock as long as the *Tadjust* > 0.
 - $SYSTIME = SYSTIME + INC\_TIME +/- 1\text{ ns}$. Add or subtract 1 ns is defined by *TIMADJ.Sign* (where '0' means Add and '1' means Subtract).

- **$TADJL = TADJL - 1 \text{ ns}$**

— As shown above, the time adjustment might take multiple clocks. Software might write a new value to the TIMADJ register before the hardware completed the previous adjustment. In such a case, the new value written by software, overrides the above equation. If such a race is not desired, the software could check that the previous adjustment is completed by one of the following methods:

- Wait enough time before accessing the TIMADJ register which guarantees that the previous update procedure is completed.
- Poll the matched TSICR.TADJ flag which is set by the hardware when the update procedure is completed.

7.7.3.2 Time Stamping Mechanism

The time stamping logic is located as close as possible to the PHY. Figure 7-38 shows the exact point in time where the time value is captured by the hardware relative to the packet content. This is to reduce delay uncertainties originated from implementation differences. As the time stamp is sampled at a very late phase in the data path, the E610 does not insert it to the transferred packet. Instead, the E610 supports the two-step operation as follows for Tx and Rx.

Note: As the time stamping logic is located at the MAC/PHY interface, there is no timestamping of packets when MAC loopback is activated.

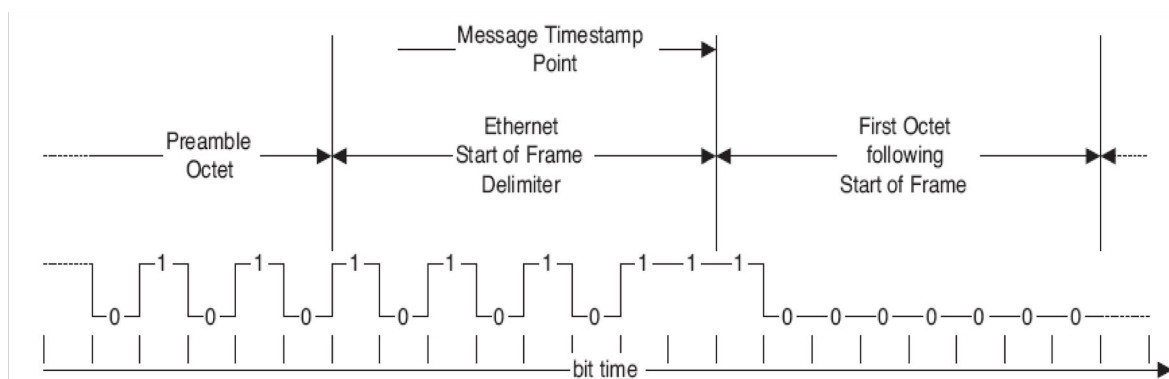


Figure 7-38. Time Stamp Point

7.7.3.2.1 Single Tx Time Stamping

The time stamp logic is activated if enabled by the `TSYNCTXCTL.EN` bit and the time stamp bit in the packet descriptor is set. In this case, hardware captures the packet's transmission time in the `TXSTMP` and `TXSTMPH` registers. Software is responsible to read the transmission time and append it in the `Folow_Up` packet as shown in Figure 7-35.

7.7.3.2.2 Single Rx Time Stamping

On the Rx, the logic parses the traversing frame. If it is matching the message type defined in `RXMTRL` register and the `TSYNCRXCTL.TYPE` field, and timestamping is enabled by the `TSYNCRXCTL.EN` bit the reception time stamp is stored in the `RXSTMP` and `RXSTMPH` registers. In addition, the TS status bits is reported in the Rx descriptor to identify that a time stamp was taken for this packet (stored in the `RXSTMP` and `RXSTMPH` registers).

Note: The time stamp values are locked in the RXSTMPL and RXSTMPH registers until software accesses them. As long as software does not read these registers, hardware does not capture the time stamp of further Rx packets. Software is expected to read these registers as soon as it processes a packet with the 'TS' indication in the Rx descriptor.

Note: If `TSYNCRXCTL.TYPE = 4` (sample time stamp of all packets), the timestamps are not locked in the RXSTMPL and RXSTMPH registers, and each new packet timestamp is stored in these registers. This mode is most likely usable only for some test environment cases.

7.7.3.2.3 Multiple Rx Time Stamping

Packets that are identified to be time stamped by hardware are also indicated by the *TS* or *TSIP* flags in the receive descriptors. If the *TS* flag is set, the packet reception time is sampled by the hardware in the RXSTMPL/H registers (see Single Rx Time Stamping above). These registers are locked till the software reads its value. If the *TSIP* flag(s) in the TSYNCRXCTL register are set, the packet reception time is posted to the packet buffer in host memory. For more information about posting the time stamp in the receive buffer refer to [Section 7.1.6.2](#).

7.7.4 Hardware PTP Elements

All PTP hardware is initialized as defined in the registers section upon MAC reset. The PTP logic is enabled if the `TSAUXC.Disable systime` flag is cleared.

The PTP logic includes multiple registers larger than 32 bits which are indicated as xxxL (Low portion - LS) and xxxH (High portion - MS). When software accesses these registers (either read or write) it should access first the xxxL register (LS) and only then the xxxH register (MS). Accessing the xxxH might impact the hardware functionality which should be triggered only after both portions of the register are valid.

7.7.4.1 Target Time

The two sets of target time registers TRGTTIMEL0/H0 and TRGTTIMEL1/H1 enable generating a time triggered event to external hardware using one of the SDP pins according to the setup defined in the TSSDP and TSAUXC registers. Each target time register is structured the same as the SYSTIMEL/H registers. If the value of SYSTIMEL/H is equal or larger than the value of the TRGTTIMEL/H registers, a change in level or a pulse is generated on the matched SDP outputs. For any of the "Target Time" functions, the selected SDP must be defined upfront as an output pin. Selecting an SDP that is set as an input pin is erroneous setting that is ignored by the device. Furthermore, the selected SDP should be programmed upfront to "native" mode. If SDP1 is selected then the `SDP1_FUNCTION` flag should be cleared.

7.7.4.1.1 SYSTIM Synchronized Level Change Generation on SDP Pins

To generate a level change on one of the SDP pins when System Time (SYSTIM) reaches a pre-defined value, the driver should do the following:

1. Select a specific SDP pin by setting the `TSSDP.TS_SDPx_EN` flag to 1b (where 'x' is 0, 1, 2 or 3).
2. Assign a target time register to the selected SDP by setting the `TSSDP.TS_SDPx_SEL` field to 00b or 01b if level change should occur based on TRGTTIMEL0/H0 or TRGTTIMEL1/H1, respectively.
3. Program the selected target time TRGTTIMELx/Hx to the required event time (where 'x' is 0 or 1).

4. TSAUXC register settings: Enable the selected target time register by setting the matched *EN_TTx* (where 'x' is 0 or 1) flags. Set the required functionality to "Level Change" by clearing the *PLSG0* flag.
5. When the SYSTIMEL/H registers become equal or larger than the selected TRGTTIML/H registers, the selected SDP changes its output level.

7.7.4.1.2 SYSTIM Synchronized Pulse Generation on SDP Pins

An output pulse can be generated by using one of the target time registers to define the beginning of the pulse and the other target time registers to define the pulse completion time. To generate a pulse on one of the SDP pins when System Time (SYSTIM) reaches a pre-defined value, the driver should do the following:

1. Select a specific SDP pin by setting the *TSSDP.TS_SDPx_EN* flag to 1b (where 'x' is 0, 1, 2 or 3).
2. Set *TSSDP.TS_SDPx_SEL* field to 00b to define that the TRGTTIMEL0/H0 register defines the start of pulse time, and the TRGTTIMEL1/H1 register defines the end of pulse time.
3. Program the selected target time TRGTTIMELx/Hx to the required event time (while 'x' is 0b or 1b). Note that the TRGTTIMEL0/H0 should be set to a lower value than TRGTTIMEL1/H1.
4. TSAUXC register settings: Enable the selected target time register by setting the matched *EN_TTx* (where 'x' is 0 or 1) flags. Set the required functionality to "Start of Pulse" by setting the *PLSG0* flag to 1b.
5. When the SYSTIMEL/H registers becomes equal or larger than the TRGTTIMEL0/H0 registers the selected SDP changes its level. Then, when the SYSTIMEL/H registers becomes equal or larger than TRGTTIMEL0/H1 registers (that define the trailing edge of the pulse), the selected SDP changes its level back.

7.7.4.1.3 Synchronized Output Clock on SDP Pins

The E610 supports driving a programmable Clock on the SDP pins (up to two output clocks). The output clocks generated are synchronized to the global System time registers (SYSTIM). The Target Time registers (TRGTTIMEL0/H0 or TRGTTIMEL2/H1) can be used for the clock output generation. To start an clock output on one of the SDP pins when System Time (SYSTIM) reaches a pre-defined value, the driver should do the following:

1. Select a specific SDP pin by setting the *TSSDP.TS_SDPx_EN* flag to 1b (where 'x' is 0, 1, 2 or 3).
2. Select the target time register for a selected SDP, by setting the *TSSDP.TS_SDPx_SEL* field to 10b or 11b if output clock should occur based on TRGTTIMEL0/H0 or TRGTTIMEL2/H1 respectively.
3. Program the matched *FREQOUT0/1* register to define clock half cycle time.
4. TSAUXC register settings:
 - Enable the selected target time register by setting the matched *EN_TTx* flag.
 - Enable the clock operation by setting the matched *EN_CLK0/1* bit to 1b.
 - If the output clock should start at a specific time, the *ST0/1* flag should be set to 1b and the matched TRGTTIMELx/Hx should be set to the required start time.

5. The clock out drives initially a logical '0' level on the selected SDP. If the TSAUXC.ST0/1 flag is cleared, it happens instantly when setting the TSAUXC.EN_CLK0/1 bit. Otherwise it happens when SYSTIM is equal or larger than the TRGTTIM. Since then, the hardware repeats endlessly the following two steps:
 - Increment the used TRGTTIMELx/Hx by *FREQOUT*.
 - When SYSTIM is equal or larger than the TRGTTIM, the SDP reverts its output level.

7.7.4.2 Time Stamp Events

Note: When clearing TSAUXC.EN_CLK0/1 while TSAUXC.EN_TT0/1 was set to generate interrupts, clear the matching TSAUXC.EN_TT0/1 too to avoid one unexpected toggle.
Yoram: Please check the functionality described above + the functionality of the DIS_TS_CLEAR flag relating repeated functionality and the interrupt.

Upon a change in the input level of one of the SDP pins that was configured to detect Time stamp events using the TSSDP register or upon a setting of one of the TSAUXC.SAMP_AUTx fields, a time stamp of the SYSTIM registers is captured into one of the two auxiliary time stamp registers (AUXSTMPLO/H0 or AUXSTMPHI/H1). Software enables the timestamp of input event as follows:

1. Define the sampled SDP on AUX time 'x' ('x' = 0b or 1b) by setting the TSSDP.AUXx_SDP_SEL field while setting the matched TSSDP.AUXx_TS_SDP_EN bit to 1b.
2. Set also the TSAUXC.EN_TSx bit ('x' = 0b or 1b) to 1b to enable "timestamping".

Following a transition on the selected SDP, the hardware does the following:

1. The SYSTIM registers (low and high) are latched to the selected AUXSTMP registers (low and high).
2. The selected *AUTT0* or *AUTT1* flags are set in the TSICR and TSAUXC registers. If the *AUTT* interrupt is enabled by the TSIM register, and the 1588 interrupts are enabled by the *TIMESYNC* flag in the ICR register, an interrupt is asserted as well. After the hardware reports that an event time was latched, the software should read the latched time in the selected AUXSTMP registers. Software should read first the Low register and only then the High register. Reading the high register clears the relevant TSAUC.AUTT_x field and releases the registers to sample a new event.

The software device driver may initiate a sampling of the current time by setting one of the TSAUXC.SAMP_AUTx fields. When one of these bits is written the hardware The SYSTIM registers (low and high) are latched to the selected AUXSTMP registers (low and high) and the selected *AUTT0* or *AUTT1* flags are set in the TSAUXC register. Once the device driver reads the selected AUXSTMP registers as described above, the hardware clears the relevant TSAUXC.AUTT_x field and releases the registers to sample a new event.

7.7.5 PTP Interrupts

PTP related interrupts can be generated by programming the TSICR and TSIM registers. The TSICR register logs the interrupt cause and the TSIM register enables masking specific TSICR bits. Occurrence of a PTP interrupt sets the ICR.TIME_SYNC interrupt bit.

7.7.6 PTP Packet Structure

The PTP implementation supports both the 1588 V1 and V2 PTP frame formats. The V1 structure can come only as UDP payload over IPv4 while the V2 can come over L2 with its EtherType or as a UDP payload over IPv4 or IPv6. The 802.1AS uses only the layer 2 V2 format. Note that PTP frame structure over UDP is not supported in the E610 for IP tunneling packets.

Offset in Bytes	V1 Fields	V2 Fields		
Bits	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0		
0	versionPTP	transportSpecific ¹	messageType	
1		Reserved	versionPTP	
2	versionNetwork	messageLength		
3				
4	Subdomain	SubdomainNumber		
5		Reserved		
6		flags		
7				
8		Correction Field		
9				
10				
11				
12				
13				
14				
15				
16		reserved		
17				
18				
19				
20	messageType	Source Port ID		
21	Source communication technology			
22	Sourceuuid			
23				
24				
25				
26				
27				
28	sourceportid			
29				
30	sequenceId	sequenceId		
31				

Offset in Bytes	V1 Fields	V2 Fields
Bits	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
32	control	control
33	reserved	logMessagePeriod
34	falgs	n/a
35		

1. Should all be zero.

Note: Only the fields with the bold italic format colored red are of interest to hardware.

Ethernet (L2)	VLAN (Optional)	PTP EtherType	PTP message
Ethernet (L2)	IP (L3)	UDP	PTP message

When a PTP packet is recognized (by EtherType or UDP port address) on the Rx side, the version should be checked if it is V1, then the control field at offset 32 should be compared to message field in the RXMTRL register described in [Section 8.2.2.19.14](#). Otherwise, the byte at offset 0 should be used for comparison to the rest of the needed field are at the same location and size for both V1 and V2.

Enumeration	Value
PTP_SYNC_MESSAGE	0
PTP_DELAY_REQ_MESSAGE	1
PTP_FOLLOWUP_MESSAGE	2
PTP_DELAY_RESP_MESSAGE	3
PTP_MANAGEMENT_MESSAGE	4
reserved	5-255

MessageId	Message Type	Value (hex)
PTP_SYNC_MESSAGE	Event	0
PTP_DELAY_REQ_MESSAGE	Event	1
PTP_PATH_DELAY_REQ_MESSAGE	Event	2
PTP_PATH_DELAY_RESP_MESSAGE	Event	3
Unused		4-7
PTP_FOLLOWUP_MESSAGE	General	8
PTP_DELAY_RESP_MESSAGE	General	9
PTP_PATH_DELAY_FOLLOWUP_MESSAGE	General	A
PTP_ANNOUNCE_MESSAGE	General	B
PTP_SIGNALLING_MESSAGE	General	C
PTP_MANAGEMENT_MESSAGE	General	D
Unused		E-F

If V2 mode is configured in TSAUXC register ([Section 8.2.2.19.16](#)), time stamp should be taken on PTP_PATH_DELAY_REQ_MESSAGE and PTP_PATH_DELAY_RESP_MESSAGE for any value in the message field in the RXMTRL register described at [Section 8.2.2.19.14](#).

Note: A PTP over UDP packet encapsulated in a VXLAN or NVGRE tunnel is still identified as a PTP packet.

7.7.6.1 PTP Packets Identification Configuration

Table 7-61 summarize the setting needed to identify PTP packets.

Table 7-61. Enabling Receive Timestamp

Functionality	Register	Field	Setting Options
Enable receive timestamp in register	TSYNCRXCTL	En	En = 1b (must be set in all the following options).
Sampled V1 Control value	RXMTRL	CTRLT	The <i>CTRLT</i> defines the recognized V1 <i>Control</i> field. This field must be defined if V1 packets recognition is required. 0x0 = PTP_SYNC_MESSAGE
Sampled V2 MessageType value	RXMTRL	MSGT	The <i>MSGT</i> defines the recognized V2 <i>MessageType</i> field. This field must be defined if V2 packets recognition is required. 0x0 = Sync
Enable all packets for timestamp	TSYNCRXCTL	Type	Type equals to 100b enables sampling all packets. Useful only when posting the timestamp to the packet buffer in host memory, enabled per queue by the <i>SRRCTL[n].Timestamp</i> .
Enable L2 1588 packets for timestamp sampling	TSYNCRXCTL	Type	Type equals to 000b or 010b enable V2 packets with MessageType equals to MSGT Type equals to 101b enable all V2 packets with Message Type bit 3 zero (means any event packets)
	ETQF[n]	EType Filter enable IEEE_1588_TIME_STAMP	The EType on one of the enabled ETQF registers (Filter enable is '1') should be set to the 1588 EtherType (equals to 0x88F7) and <i>IEEE_1588_TIME_STAMP</i> field should be set.
Enable 1588 packets over UDP for timestamp sampling	TSYNCRXCTL	Type	Type equals to 001b enables V1 packets with <i>Control</i> field equals to CTRLT parameter Type equals to 010b enables V2 packets with <i>MessageType</i> fields equals to MSGT parameter Type equals to 101b enables all V2 packets with <i>MessageType</i> bit 3 zero (which means any event packets)
	RXMTRL	UDPT	Defines the UDP port to use for V1 detection
Define specific receive queue for the L2 1588 packets	ETQF[n]	Rx Queue Queue Enable	Setting the "Queue Enable" on the same ETQF register as above, the receive queue is defined by the <i>Rx Queue</i> field.

7.7.7 Synchronous Ethernet - SyncE

The E610 support the ability to output a differential RX recovered clock from any port for synchronous Ethernet purpose. A complete SyncE solution should include external components on the board as a PLL, filter and muxing operation to return the clock back as the TX Reference clock. in this way, the E610 PHY transmitter clock, for all lanes, will use the same clock phase as the RX recovered clock.

7.7.7.1 Ethernet PHY Clocking

The Ethernet SerDes PLL is driven from a 156.25 MHz reference clock on the platform level.

At the platform/board level, a SyncE DPLL will perform the function of jitter attenuation and filtering of the recovered clock, provided via the 3.3V GPIOs from the E610 die. The filtered reference clock will then be fed into ETH_REFCLK_P/N, for consumption by the PLL residing in the Ethernet PHY.

In 10GBASE-T SKU, the PHY recovered clock is provided through PHY RCLK.

The RCLK pin of the device outputs a 25 MHz clock that is based on the 125, 200, 400, or 800 MHz recovered clock on the copper receive path when linked to 100BASE-TX, 1000BASE-T, 2.5GBASE-T, 5GBASE-T, or 10GBASE-T. The 125 MHz recovered clock is internally divided by 5 with 60% low and 40% high. The 200, 400, or 800 MHz recovered clock is internally divided by 8, 16, or 32, respectively, with a 50% duty cycle.

RCLK drives low when the link is down. synchronous clocking is used to get the entire system to run frequency locked with no frequency drift.

7.7.7.2 SyncE Flow

Software implements the SyncE application and configures the E610 and the DPLL using CSR based Admin Queue (AQC). EMP FW/DNL is responsible for SyncE initialization sequence and for running the software commands.

SyncE control can be set by software driver via AQC. For the supported SyncE AQ commands see [Section 3.7.5](#).

7.8 Virtualization

7.8.1 Overview

I/O virtualization is a mechanism that can be used to share I/O resources among several consumers. For example, in a virtual system, multiple operating systems are loaded and each operates as though the entire system's resources were at its disposal. However, for the limited number of I/O devices, this presents a problem because each operating system might be in a separate memory domain and all the data movement and device management has to be done by a Virtual Machine Monitor (VMM). VMM access adds latency and delay to I/O accesses and degrades I/O performance. Virtualized devices are designed to reduce the burden of the VMM by making certain functions of an I/O device shared among multiple guest operating systems or a Virtual Machine (VM), thereby allowing each VM direct access to the I/O device.

The E610 supports two modes of operations of virtualized environments:

1. Direct assignment of part of the port resources to different guest operating systems using the PCI SIG SR IOV standard. Also known as native mode or pass-through mode. This mode is referenced as **IOV mode** throughout this section.
2. Central management of the networking resources by an IOVM or by the VMM that have access rights to the NIC. Also known as software switch acceleration mode. This mode is referred to as **VMDq2 mode in this section**.

The virtualization offloads capabilities provided by the E610 apart from the replication of functions defined in the PCI SIG IOV specification are part of **VMDq2**.

A hybrid model, where part of the VMs are assigned a dedicated share of the port and the rest are serviced by an IOVM is also supported. However, in this case the offloads provided to the software switch might be more limited. This model can be used when parts of the VMs run operating systems for which VF drivers are available and thus can benefit from an IOV and others that run older operating systems for which VF drivers are not available and are serviced by an IOVM. In this case, the IOVM is assigned one VF and receives all the packets with Ethernet MAC Addresses of the VMs behind it.

The following section describes the support the E610 provides for these modes.

This section assumes a single-root implementation of IOV and no support for multi-root.

7.8.1.1 Direct Assignment Model

The direct assignment support in the E610 is built according to the following model of the software environment.

It is assumed that one of the software drivers sharing the port hardware behaves as a master driver (**Physical Function** or **PF** driver). This driver is responsible for the initialization and the handling of the common resources of the port. All the other drivers (**Virtual Function** drivers or **VF** drivers) might read part of the status of the common parts but cannot change them. The PF driver might run either in the VMM or in some service operating system. It might be part of an IOVM or part of a dedicated service operating system.

In addition, part of the non time-critical tasks are also handled by the PF driver. For example, access to CSR through the I/O space or access to the configuration space are available only through the master interface. Time-critical CSR space like control of the Tx and Rx queue or interrupt handling is replicated per VF, and directly accessible by the VF driver.

Note: In some systems with a thick hypervisor, the service operating system might be an integral part of the VMM. For these systems, each reference to the service operating system in the sections that follow refer to the VMM.

7.8.1.1.1 Rationale

The direct assignment model enables each of the VMs to receive and transmit packets with minimum of overhead. Non time-critical operations such as initialization and error handling can be done via the PF driver. In addition, it is important that the VMs can operate independently with minimal disturbance. It is also preferable that the VM interface to hardware should be as close as possible to the native interface in non-virtualized systems to minimize the software development effort.

The main time critical operations that require direct handling by the VM are:

- Maintenance of the data buffers and descriptor rings in host memory. To support this, the DMA accesses of the queues associated to a VM should be identified as such on the PCIe using a different requester ID.
- Handling of the hardware ring (tail bump and head updates)
- Interrupts handling

The capabilities needed to provide independence between VMs are:

- Per-VM reset and enable capabilities
- Tx rate control
- Allocating separate CSR space per VM. This CSR space is organized as close as possible to the regular CSR space to enable sharing of the base driver code.

Note: The rate control and VF enable capabilities are controlled by the PF.

7.8.1.2 System Overview

The following drawings show the various elements involved in the I/O process in a virtualized system. [Figure 7-39](#) shows the flow in software VMDq2 mode and [Figure 7-40](#) shows the flow in IOV mode.

This section assumes that in IOV mode, the driver on the guest operating system is aware that it operates in a virtual system (para-virtualized) and there is a channel between each of the VM drivers and the PF driver allowing message passing such as configuration request or interrupt messages. This channel can use the mailbox system implemented in the E610 or any other means provided by the VMM vendor.

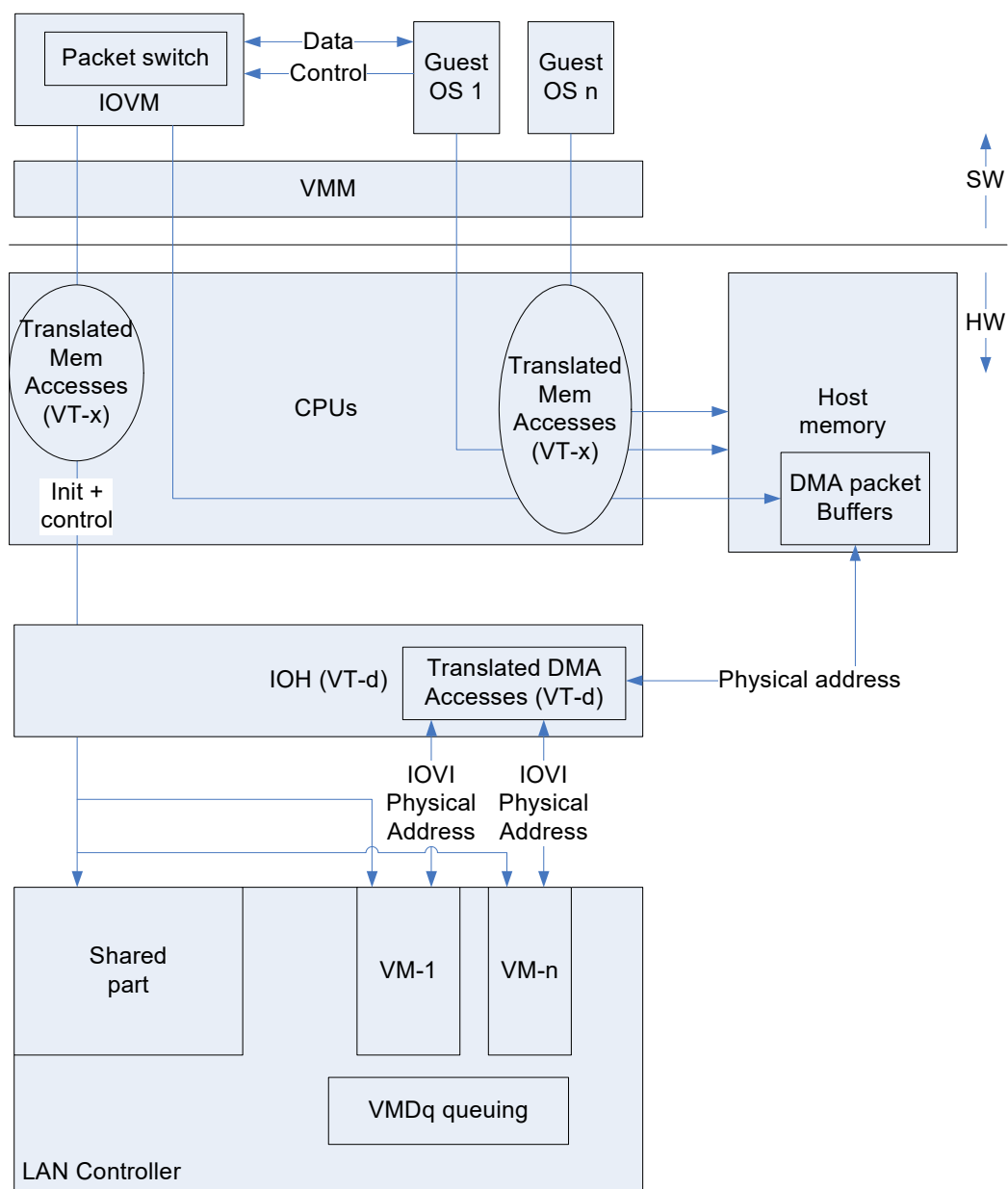


Figure 7-39. System Configuration for VMDq2 Mode

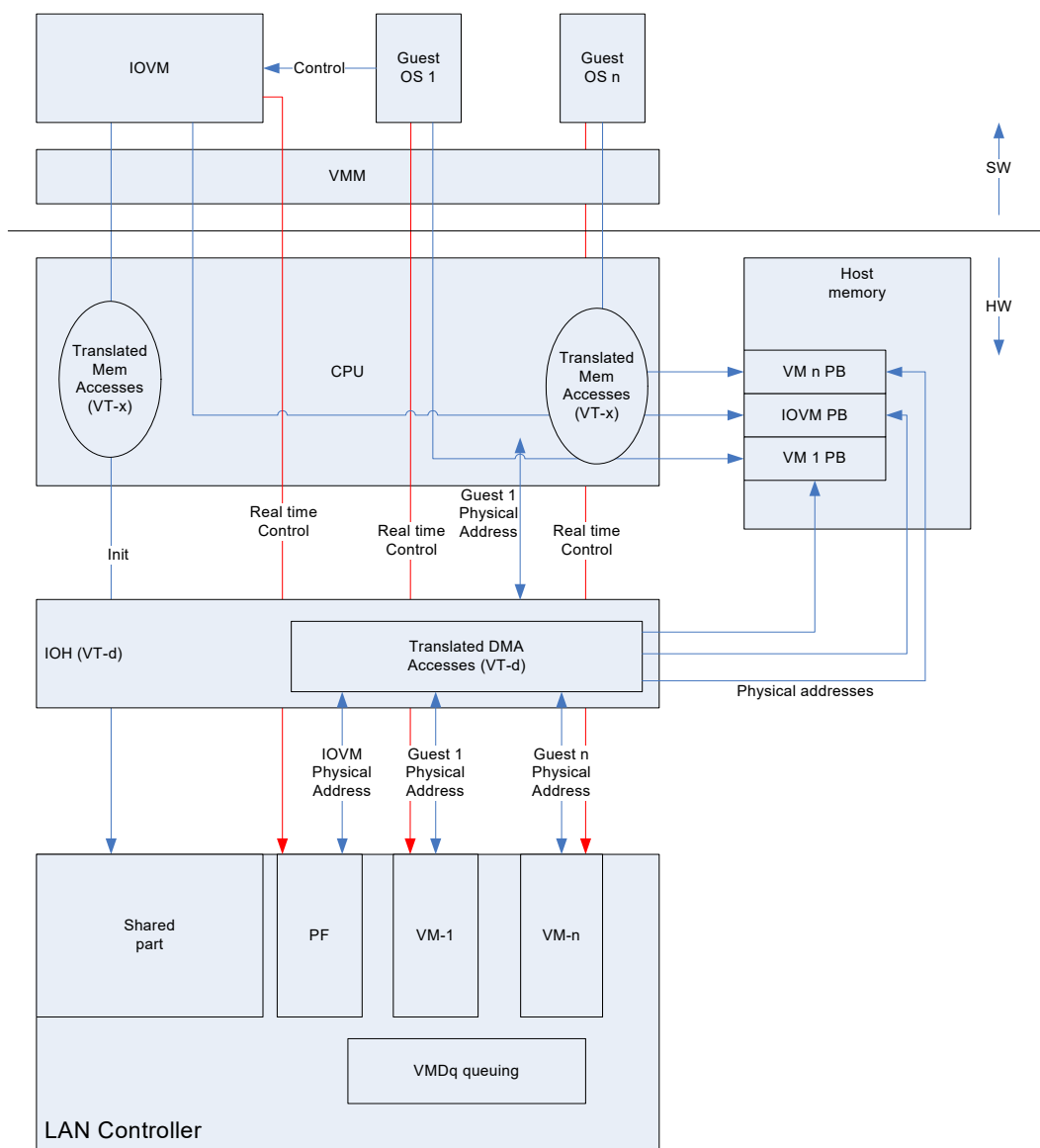


Figure 7-40. System Configuration for IOV Mode

7.8.2 PCI-SIG SR-IOV Support

7.8.2.1 SR-IOV Concepts

SR-IOV defines the following entities in relation to I/O virtualization:

- **Virtual Image (VI)** — Part of the I/O resources are assigned to a VM.
- **I/O Virtual Intermediary (IOVI) or I/O Virtual Machine (IOVM)** — A special VM that owns the physical device and is responsible for the configuration and managing the physical device.
- **Physical function (PF)** — A function representing a physical instance — One port for the E610. The PF driver is responsible for the configuration and management of the shared resources in the function.
- **Virtual Function (VF)** — A part of a PF assigned to a VI.

7.8.2.2 Configuration Space Replication

The SR-IOV specification defines a reduced configuration space for the virtual functions. Most of the PCIe configuration of the VFs comes from the PF.

This section describes the expected handling of the different parts of the configuration space for virtual functions. It deals only with the parts relevant to the E610.

Details of the configuration space for virtual functions can be found in [Section 9.3](#).

7.8.2.2.1 Legacy PCI Configuration Space

The legacy configuration space is allocated to the PF only and emulated for the VFs. A separate set of BARs and one bus master enable bit is allocated in the SR-IOV capability structure in the PF and is used to define the address space used by the entire set of VFs.

All the legacy error reporting bits are emulated for the VF. See [Section 7.8.2.4](#) for details.

7.8.2.2.2 Memory BARs Assignment

The SR-IOV specification defines a fixed stride for all the VF BARs, so that each VF can be allocated part of the memory BARs at a fixed stride from a basic set of BARs. In this method, only two decoders per replicated BAR per PF are required and the BARs reflected to the VF are emulated by the VMM.

The only BARs that are useful for the VFs are BAR0 and BAR3, so only those are replicated. [Table 7-62](#) lists the existing BARs and the stride used for the VFs:

Table 7-62. BARs in the E610 (64-bit BARs)

BAR	Type	Usage	Requested Size per VF (=Stride)
0, 1	Mem	CSR space	Maximum (16 KB, page size). For page size see Section 9.2.4.4.8 for more details.
2	n/a	Not used	N/A
3	n/a	Not used	N/A
4, 5	Mem	MSI-X	Maximum (16 KB, page size).

BAR0 of the VFs are a sparse version of the original PF BAR and include only the register relevant to the VF. For more details see [Section 7.8.2.7](#).

Figure 7-41 shows the different BARs in an IOV-enabled system:

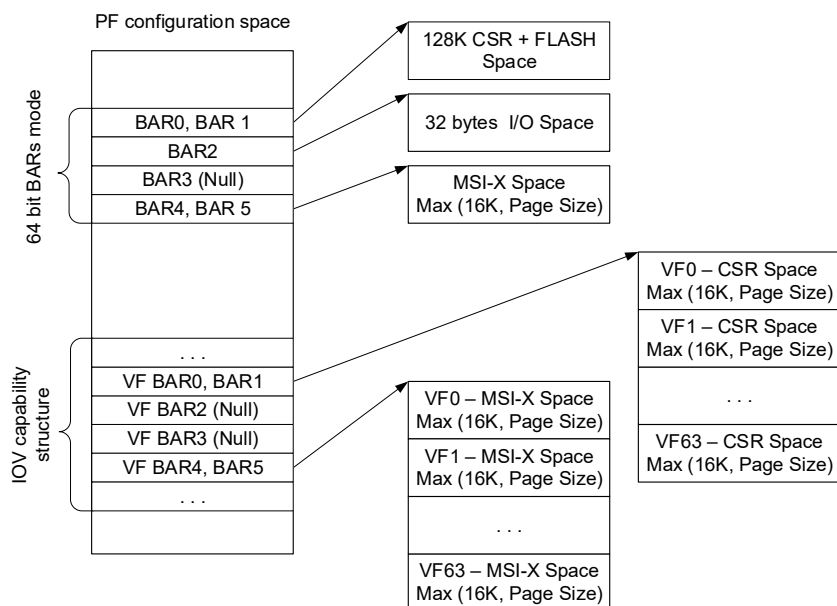


Figure 7-41. BARs in an IOV-enabled System

7.8.2.2.3 PCIe Capability Structure

The PCIe capability structure is shared between the PF and the VFs. The only relevant bits that are replicated are:

- Transaction pending.
- Function Level Reset (FLR). See [Section 7.8.2.3](#) for details.

7.8.2.2.4 MSI and MSI-X Capabilities

Both MSI and MSI-X are implemented in the E610. MSI-X vectors can be assigned per VF. MSI is not supported for the VFs.

See [Table 9-17](#) for more details of the MSI-X and PBA tables implementation.

7.8.2.2.5 VPD Capability

VPD is implemented only once and is accessible only from the PF.

7.8.2.2.6 Power Management Capability

The E610 does not support power management per VF. The power management registers exist for each VF, but only the D0 power state is supported.

7.8.2.2.7 Serial ID

Serial ID capability is not exposed in VFs.

7.8.2.2.8 Error Reporting Capabilities (Advanced and Legacy)

All the bits in this capability structure are implemented only for the PF. Note that the VMs see an emulated version of this capability structure. See [Section 7.8.2.4](#) for details.

7.8.2.3 FLR Capability

The *FLR* bit is required per VF. Setting of this bit resets only a part of the logic dedicated to the specific VF and does not influence the shared part of the port. This reset should disable the queues, disable interrupts and the stop receive and transmit process per VF.

Setting the PF *FLR* bit resets the entire function.

7.8.2.4 Error Reporting

Error reporting includes legacy error reporting and Advanced Error Reporting (AER) or role-based capability.

The legacy error management includes the following functions:

- **Error capabilities enablement** — These are set by the PF for all the VFs. Narrower error reporting for a given VM can be achieved by filtering of the errors by the VMM. This includes:
 - SERR# Enable
 - Parity Error Response
 - Correctable Reporting Enable
 - Non-Fatal Reporting Enable
 - Fatal Reporting Enable
 - UR Reporting Enable
- **Error status in the configuration space** — These should be set separately for each VF. This includes:
 - Master Data Parity Error
 - Signaled Target Abort
 - Received Target Abort
 - Master Abort
 - SERR# Asserted
 - Detected Parity Error
 - Correctable Error Detected
 - Non-Fatal Error Detected
 - Unsupported Request Detected

AER capability includes the following functions:

- **Error capabilities enablement** — The *Error Mask*, and *Severity* bits are set by the PF for all the VFs. Narrower error reporting for a given VM can be achieved by filtering of the errors by the VMM. These includes:
 - Uncorrectable Error Mask Register
 - Uncorrectable Error Severity Register
 - Correctable Error Mask Register
 - ECRC Generation Enable
 - ECRC Check Enable
- **Non-Function Specific Errors Status in the configuration space**
 - Non-Function Specific Errors are logged in the PF
 - Error logged in one register only
 - VI avoids touching all VFs to clear device level errors
 - The following errors are not function specific
 - All Physical Layer errors
 - All Link Layer errors
 - ECRC Fail
 - UR, when caused by no function claiming a TLP
 - Receiver Overflow
 - Flow Control Protocol Error
 - Malformed TLP
 - Unexpected Completion
- **Function Specific Errors Status in the configuration space**
 - Allows Per-VF error detection and logging
 - Help with fault isolation
 - The following errors are function specific:
 - Poisoned TLP received
 - Completion Timeout
 - Completer Abort
 - UR, when caused by a function that claims a TLP
 - ACS Violation
- **Error logging** — Each VF has it's own header log.
- **Error messages** — To ease the detection of the source of the error, the error messages should be emitted using the requester ID of the VF in which the error occurred.

7.8.2.5 Alternative Routing ID (ARI) and IOV Capability Structures

To allow more than eight functions per end point without requesting an internal switch, as usually needed in virtualization scenarios, the PCI-SIG defines the ARI capability structure. This is a new capability that enables an interpretation of the *Device* and *Function* fields as a single identification of a function within the bus. In addition, a new structure used to support the IOV capabilities reporting and control is defined. Both structures are described in sections [Section 9.2.4.3](#) and [Section 9.2.4.4](#). Refer to the following section for details on the Requester ID (RID) allocation to VFs.

7.8.2.6 RID Allocation

RID allocation of the VF is done using the *Offset* field in the IOV structure. This field should be replicated per VF and is used to do the enumeration of the VFs.

Each PF includes an offset to the first associated VF. This pointer is a relative offset to the Bus/Device/Function (BDF) of the first VF. The *Offset* field is added to PF's requester ID to determine the requester ID of the next VF. An additional field in the IOV capability structure describes the distance between two consecutive VF's requester IDs.

7.8.2.6.1 Bus Device Function Layout

7.8.2.6.1.1 ARI Mode

ARI allows interpretation of the device ID part of the RID as part of the function ID inside a device. Thus, a single device can span up to 256 functions. To ease the decoding, the least significant bit of the function number points to the physical port number. The *Next* bits indicate the VF number. [Table 7-63](#) lists the VF RIDs.

The layout of RID's used by the E610 is reported to the operating system via the PCIe IOV capability structure. See [Section 9.2.4.4.6](#).

Table 7-63. RID per VF — ARI Mode

Port	VF#	B,D,F	Binary	Notes
0	PF	B,0,0	B,00000,000	PF
1	PF	B,0,1	B,00000,001	PF
0	1	B ,16,0	B,10000,000	Offset to first VF from PF is 128.
1	1	B ,16,1	B,10000,001	
0	2	B ,16,2	B,10000,010	
1	2	B ,16,3	B,10000,011	
0	3	B ,16,4	B,10000,100	
1	3	B ,16,5	B,10000,101	
...				
0	64	B ,31,6	B,11111,110	
1	64	B ,31,7	B,11111,111	Last

7.8.2.6.1.2 Non-ARI Mode

When ARI is disabled, non-zero devices in the first bus cannot be used, thus a second bus is needed to provide enough RIDs. In this mode, the RID layout is as follows:

Table 7-64. RID per VF — Non-ARI Mode

Port	VF#	B,D,F	Binary	Notes
0	PF	B,0,0	B,00000,000	PF
1	PF	B,0,1	B,00000,001	PF
0	0	B+1 ,16,0	B+1,10000,000	Offset to first VF from PF is 384.
1	0	B+1 ,16,1	B+1,10000,001	
0	1	B+1 ,16,2	B+1,10000,010	
1	1	B+1 ,16,3	B+1,10000,011	
0	2	B+1 ,16,4	B+1,10000,100	
1	2	B+1 ,16,5	B+1,10000,101	
...				
0	63	B+1 ,31,6	B+1,11111,110	
1	63	B+1 ,31,7	B+1,11111,111	Last

Note: When the device ID of a physical function changes (because of LAN disable or LAN function select settings), the VF device IDs changes accordingly.

7.8.2.7 Hardware Resources Assignment

The main resources to allocate per VM are queues and interrupts. The assignment is static. If a VM requires more resources, it might be allocated to more than one VF. In this case, each VF gets a specific Ethernet MAC Address/VLAN tag to enable forwarding of incoming traffic. The two VFs are then teamed in software.

7.8.2.7.1 PF Resources

A possible use of the PF is for a configuration setting without transmit and receive capabilities. In this case, it is not allocated to any queues but is allocated to one MSI-X vector.

The PF has access to all the resources of all VMs, but it is not expected to make use of resources allocated to active VFs.

7.8.2.7.2 Assignment of Queues to VF

See [Section 7.2.1.2.1](#) for allocating Tx queues.

See [Section 7.1.3.2](#) for allocating Rx queues.

[Table 7-65](#) lists the Tx and Rx queues to VF allocation.

Table 7-65. Queue to VF Allocation

VF	Queues in 16 VMs Mode	Queues in 32 VMs Mode	Queues in 64 VMs Mode
0	0-7	0-3	0-1
1	8-15	4-7	2-3
...	...	...	...
15	120-127	...	...
...		...	...
31		124-127	...
...			...
63			126-127

7.8.2.7.3 Assignment of MSI-X Vector to VF

See [Section 7.3.4.3](#) for allocating MSI-X vectors in IOV mode.

7.8.2.8 CSR Organization

CSRs can be divided into three types:

- Global Configuration registers that should be accessible only to the PF — For example, link control and LED control. These types of registers also include all of the debug features such as the mapping of the packet buffers and is responsible for most of the CSR area requested by the E610. This includes per VF configuration parameters that can be set by the PF without performance impact.
- Per-VF parameters — For example, per VF reset, interrupt enable, etc. Multiple instances of these parameters are used only in an IOV system and only one instance is needed for non IOV systems.
- Per-queue parameters that should be replicated per queue — For example, head, tail, Rx buffer size, TPH tag, etc. These parameters are used by both a VF in an IOV system and by the PF in a non-IOV mode.

To support IOV without distributing the current drivers operation in legacy mode, the following method is used:

- The PF instance of BAR0 continues to contain the legacy and control registers. It is accessible only to the PF. The BAR enables access to all the resources including the VF queues and other VF parameters. However, it is expected that the PF driver does not access these queues in IOV mode.
- The VF instances of BAR0 provide control on the VF specific registers. These BARs have the same mapping as the original BAR0 with the following exceptions:
 - Fields related to the shared resources are reserved.
 - The queues assigned to a VF are mapped at the same location as the first same number of queues of the PF.
- Assuming some backward compatibility is needed for IOV drivers, The PF/VF parameters block should contain a partial register set as described in VF Device Registers section.

7.8.2.9 SR IOV Control

To control the IOV operation, the physical driver is provided with a set of registers. These include:

- The mailbox mechanism described in the next section.
- The switch and filtering control registers described in [Section 7.8.3.9](#).
- PFVFLREC register indicating that a VFLR reset occurred in one of the VFs (bitmap).

7.8.2.9.1 VF-to-PF Mailbox

The VF drivers and the PF driver require some means of communication between them. This channel can be used for the PF driver to send status updates to the VFs (such as link change, memory parity error, etc.) or for the VF to send requests to the PF (add to VLAN).

Such a channel can be implemented in software, but requires enablement by the VMM vendors. To avoid the need for such an enablement, the E610 provides such a channel that enables direct communication between the two drivers.

The channel consists of a mailbox. Each driver can then receive an indication (either poll or interrupt) when the other side wrote a message.

Assuming a maximum message size of 64 bytes (one cache line), a memory of 64 bytes x 64 VMs = 4 KB is provided per port. The RAM is organized as follows:

Table 7-66. Mailbox Memory

RAM Address	Function	PF BAR 0 Mapping ¹	VF BAR 0 Mapping ²
0 — 63	VF0 <-> PF	0 — 63	VF0 + MBO
64 — 127	VF1 <-> PF	64 — 127	VF1 + MBO
....			
(4 KB-64) — (4 KB-1)	VF63 <-> PF	(4 KB-64) — (4 KB-1)	VF63 + MBO

1. Relative to mailbox offset.

2. MBO = mailbox offset in VF CSR space.

In addition for each VF, the VFMailbox and PFMailbox registers are defined to coordinate the transmission of the messages. These registers contain a semaphore mechanism to enable coordination of the mailbox usage.

The PF driver can decide which VFs are allowed to interrupt the PF to indicate a mailbox message using the PFMBIMR mask register.

The following flows describe the usage of the mailbox:

Table 7-67. PF-to-VF Messaging Flow

Step	PF Driver	Hardware	VF #n driver
1	Set PFMAILBOX[n].PFU.		
2		Set PFU bit if PFMAILBOX[n].VFU is cleared.	
3	Read PFMAILBOX[n] and check that PFU bit was set. Otherwise wait and go to Step 1.		
4	Write message to relevant location in VFMBMEM.		

Table 7-67. PF-to-VF Messaging Flow [continued]

Step	PF Driver	Hardware	VF #n driver
5	Set the PFMAILBOX[n].STS bit and wait for ACK ¹ .		
6		Indicate an interrupt to VF #n.	
7			Read the message from VFMBMEM.
8			Set the VFMAILBOX[n].ACK bit.
9		Indicate an interrupt to PF.	
10	Clear PFMAILBOX[n].PFU.		

1. The PF might implement a timeout mechanism to detect non-responsive VFs.

Table 7-68. VF-to-PF Messaging Flow

Step	PF Driver	Hardware	VF #n Driver
1			Set VFMAILBOX.VFU.
2		Set VFU bit if VFMAILBOX[n].PFU is cleared.	
3			Read VFMAILBOX[n] and check that VFU bit was set. Otherwise wait and go to Step 1.
4			Write message to relevant location in VFMBMEM.
5			Set the VFMAILBOX[n].REQ bit.
6		Indicate an interrupt to PF.	
7	Read PFMBICR to detect which VF caused the interrupt.		
8	Read the adequate message from VFMBMEM.		
9	Set the PFMAILBOX[n].ACK bit.		
10		Indicate an interrupt to VF #n.	
11			Clear VFMAILBOX[n].VFU.

The content of the message is hardware independent and is determined by software.

The messages currently assumed by this specification are:

- Registration to VLAN/multicast packet/broadcast packets — A VF can request to be part of a given VLAN or to get some multicast/broadcast traffic.
- Reception of large packet — Each VF should notify the PF driver what is the largest packet size allowed in receive.
- Get global statistics — A VF can request information from the PF driver on the global statistics.
- Filter allocation request — A VF can request allocation of a filter for queuing/immediate interrupt support.
- Global interrupt indication.
- Indication of errors.

7.8.2.10 DMA

7.8.2.10.1 RID

Each VF is allocated a RID. Each DMA request should use the RID of the VM that requested it. See [Section 7.8.2.6](#) for details.

7.8.2.10.2 Sharing of the DMA Resources

The outstanding requests and completion credits are shared between all the VFs. The tags attached to read requests are assigned the same way as in a non-virtualized setting, although in VF systems tags can be re-used for different RIDs.

7.8.2.10.3 TPH

The TPH enable is common to all the devices (all PFs and VFs). Given a TPH enabled device, each VM might decide for each queue, on which type of traffic (data, headers, Tx descriptors, Rx descriptors) the TPH should be asserted and what is the CPU ID assigned to this queue.

Note: There are no plans to virtualize TPH in the IOH. Thus, the physical CPU ID should be used in the programming of the *CPUID* field.

7.8.2.11 Timers

7.8.2.11.1 TCP Timer

The TCP timer is available only to the PF. It might indicate an interrupt to the VFs via the mailbox mechanism.

7.8.2.11.2 IEEE 1588

IEEE 1588 is a per-link function and thus is controlled by the PF driver. The VMs have access to the real time clock register.

7.8.2.11.3 Free Running Timer

The free running timer is a PF driver resource the VMs can access. This register is read only to all VFs and is reset only by the PCI reset.

7.8.2.12 Power Management and Wake-Up

Power management is a PF resource and is not supported per VF.

7.8.2.13 Link Control

The link is a shared resource and as such is controllable only by the PF. This includes interface settings, speed and duplex settings, flow control settings, etc. The flow control packets are sent with the station Ethernet MAC Address stored in the NVM. The watermarks of the flow control process and the time-out value are also controllable by the PF only. In a DCB environment, the parameters of the per TC flow control are also part of the PF responsibilities.

Double VLAN is a network setting and as such should be common to all VFs.

7.8.2.13.1 Special Filtering Options

Pass bad packets is a debug feature. As such, pass bad packets is available only to the PF. Bad packets are passed according to the same filtering rules of the regular packets.

Note: Pass bad packets might cause guest operating systems to get unexpected packets. As a result, it should be used only for debug purposes of the entire system.

Receiving long packets is enabled separately per Rx queue in the RXDCTL registers. As this impacts the flow control thresholds, the PF should be made aware of the decision of all the VMs. Because of this, the setup of TSO packets is centralized by the PF and each VF might request this setting.

7.8.3 Packet Switching

7.8.3.1 Assumptions

The following assumptions are made:

- The required bandwidth for the VM-to-VM loopback traffic is low. That is, the PCIe bandwidth is not congested by the combination of the VM-to-VM and the regular incoming traffic. This case is handled but not optimized for. Unless specified otherwise, Tx and Rx packets should not be dropped or lost due to congestion caused by loopback traffic.
- If the buffer allocated for the VM-to-VM loopback traffic is full, it is acceptable to back pressure the transmit traffic of the same TC. This means that the outgoing traffic might be blocked if the loopback traffic is congested.
- The decision on local traffic is done only according to the Ethernet DA address and the VLAN tag. There is no filtering according to other parameters (IP, L4, etc.). The switch has no learning capabilities. In case of double VLAN mode, the inner VLAN is used for the switching functionality.
- The forwarding decisions are based on the receive filtering programming.
- No packet switching between TCs.
- Coexistence with TimeSync: time stamp is not sampled for any VM-to-VM loopback traffic.
- Coexistence with Double VLAN: When double VLAN is enabled by `DMATXCTL.GDV` and it is expected to transmit untagged packets by software, transmit to receive packet switching should not be enabled.

7.8.3.2 Pool Selection

Pool selection is described in the following sections. A packet might be forwarded to a single pool or replicated to multiple pools. Multicast and broadcast packets are cases of replication, as is mirroring.

The following capabilities determine the destination pools of each packet:

- 128 Ethernet MAC Address filters (RAH/RAL registers) for both unicast and multicast filtering. These are shared with L2 filtering. For example, the same Ethernet MAC Addresses are used to determine if a packet is received by the switch and to determine the forwarding destination. In “E-tag pool select” mode RAL registers bits [13:0] are used for E-tag filtering {GRP, E-CID_base} and pool selection.
- 64 shared VLAN filters (PFVLVF and PFVLVFB registers) — Each VM can be made a member of each VLAN.
- Hash filtering of unicast and multicast addresses (if the direct filters previously mentioned are not sufficient).
- Forwarding of broadcast packets to multiple pools.
- Forwarding by EtherType.
- Mirroring by pool, VLAN, or link.

Receive pool/queue allocation refer to [Section 7.1.3.2](#).

7.8.3.3 Rx Packets Switching

Rx packet switching is the second of three stages that determine the destination of a received packet. The three stages are defined in [Section 7.1.3](#).

As far as switching is concerned, it does not matter whether the E610’s virtual environment operates in IOV mode or in VMDq2 mode.

When operating in replication mode, broadcast and multicast packets can be forwarded to more than one pool, and is replicated to more than one Rx queue. Replication is enabled by the *Rpl_En* bit in the PFVTCTL register.

Note: For the following algorithms packets with no E-tag in *PFVTCTL.POOLING_MODE* = 01b (E-tag mode), or all packets in *PFVTCTL.POOLING_MODE* = 00b (MAC mode) are defined as packets with no external tags.

7.8.3.3.1 Replication Mode Enabled

When replication mode is enabled, each broadcast/multicast packet can go to more than one pool. Finding the pool list of any packet is provided in the following steps:

1. **Exact unicast or multicast match** — If there is a match in one of the exact filters (RAL/RAH), for unicast or multicast packets, use the MAC Pool Select Array (MPSAR[n]) bits as a candidate for the pool list. Note that MPSAR[n] must not enable more than one pool for unicast RAL/RAH filters. The compared field differs according to the filtering mode (*PFVTCTL.POOLING_MODE*):
 - a. If *POOLING_MODE* = 00b (MAC mode) or for packets with no external tags, the Destination MAC Address is compared to {RAH[15:0], RAL[31:0]} for all the registers for which RAH.ADTYPE = 0 (MAC)

- b. If *POOLING_MODE* = 01b (E-tag mode) and the packet has an E-tag, the E-tag E-PID (as extracted from offset 47:34, starting from the EtherType, in the tag identified by the ETAG_ETYPE register) is compared to RAL[13:0] for all the registers for which RAH.ADTYPE = 1 (E-tag)
2. **PFVFRE** — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list.
3. **Broadcast** — If the packet is a broadcast packet with no external tag, add pools for which their PFVML2FLT.BAM bit (*Broadcast Accept Mode*) is set.
4. **Unicast hash** — If the packet is a unicast packet with no external tag, and the prior steps yielded no pools, check it against the Unicast Hash Table (*PFUTA*). If there is a match, add pools for which their PFVML2FLT.ROPE bit (*Accept Unicast Hash*) is set.
5. **Multicast hash** — If the packet is a multicast packet with no external tag and the prior steps yielded no pools, check it against the Multicast Hash Table (*MTA*). If there is a match, add pools for which their PFVML2FLT.ROMPE bit (*Receive Multicast Packet Enable*) is set.
6. **Multicast promiscuous** — If the packet is a multicast packet with no external tag, take the candidate list from prior steps and add pools for which their PFVML2FLT.MPE bit (*Multicast Promiscuous Enable*) is set.
7. **Unicast Promiscuous** - If the packet is a unicast packet with no external tag, take the candidate list from prior steps and add pools for which their PFVML2FLT.UPE bit (*Unicast Promiscuous Enable*) is set.
8. **VLAN groups** — This step is relevant only when VLAN filtering is enabled by the VLNCTRL.VFE bit.
 - a. Tagged packets: enable only pools in the packet's VLAN group as defined by the VLAN filters — PFVLVF[n] and their pool list — PFVLVFB[n] or pools for which the PFVML2FLT.VPE (*VLAN Promiscuous Enable*) is set.
 - b. Untagged packets: enable only pools with their PFVML2FLT.AUPE bit set.
 - c. If there is no match, the pool list should be empty.
- Note:** In a VLAN network, un-tagged packets are not expected. Such packets received by the switch should be dropped, unless their destination is a virtual port set to receive these packets. The setting is done through the PFVML2FLT.AUPE bit. It is assumed that VMs for which this bit is set are members of a default VLAN and thus only MAC queuing is done on these packets.
9. **Default Pool** — If the pool list is empty at this stage and the PFVTCTL.DIS_DEF_POOL bit is cleared, set the default pool bit in the target pool list (from PFVTCTL.DEF_PL).
10. **EtherType filters** — If one of the EtherType filters (ETQF) is matched by the packet and pooling action is requested and the *Pool Enable* bit in the ETQF is set, the pool list is set to the pool pointed to by the filter.
11. **Filter Local Packets (source address pruning)** - The pruning operation depends on the filtering mode (PFVTCTL.POOLING_MODE):
 - a. If *POOLING_MODE* = 00b (MAC mode) or for packets with no external tag, the Source MAC Address is compared to {RAH[15:0], RAL[31:0]} for all the registers for which RAH.ADTYPE = 0 (MAC).
 - b. If *POOLING_MODE* = 01b (E-tag mode) and the packet has an E-tag, {00,E-tag ingress-E-CID_base} (as extracted from offset 31:20, starting from the EtherType, in the tag identified by the ETAG_ETYPE register) is compared to RAL[13:0] for all the registers for which RAH.ADTYPE = 1 (E-tag).

If there is a match to one of the RAL[n]/RAH[n] clear all the pools that are set in both the matching MPSAR[n] and the PFFLP from the pool list. This prunes multicast packets from getting back to the sender in VEPA mode.

12. **PFVFRE** — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list. The PFVFRE register blocks reception by a VF while the PF configures its registers.
13. **Mirroring** — Each of the four mirroring rules adds its destination pool (PFMRCTL.MP) to the pool list if the following applies:
 - a. **Pool mirroring** — PFMRCTL.VPME is set and one of the bits in the pool list matches one of the bits in the PFMRVM register.
 - b. **VLAN port mirroring** — PFMRCTL.VLME is set and the index of the VLAN of the packet in the PFVLVF table matches one of the bits in the PFMRVLAN register.
 - c. **Uplink port mirroring** — PFMRCTL.UPME is set, the pool list is not empty.
14. **PFVFRE** — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list. The PFVFRE register blocks reception by a VF while the PF configures its registers. Note that this stage appears twice to handle mirroring cases.

7.8.3.3.2 Replication Mode Disabled

When replication mode is disabled, software should take care of multicast and broadcast packets and check which of the VMs should get them. In this mode, the pool list of any packet always contains one pool only according to the following steps:

1. **Exact unicast or multicast match** — The compared field differs according to the filtering mode (PFVTCTL.POOLING_MODE):
 - a. If *POOLING_MODE* = 00b (MAC mode) or for packets with no external tags, the Destination MAC Address is compared to {RAH[15:0], RAL[31:0]} for all the registers for which RAH.ADTYPE = 0 (MAC)
 - b. If *POOLING_MODE* = 01b (E-tag mode) and the packet has an E-tag, the E-tag E-PID (as extracted from offset 47:34, starting from the EtherType, in the tag identified by the ETAG_ETYPE register) is compared to RAL[13:0] for all the registers for which RAH.ADTYPE = 1 (E-tag)

If the compared field matches one of the exact filters (RAL/RAH), use the MAC Pool Select Array (MPSAR[n]) bits as a candidate for the pool list. Note that MPSAR[n] must not enable more than one pool for unicast RAL/RAH filters.

2. **PFVFRE** — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list.
3. **Unicast hash** — If the packet is a unicast packet with no external tag, and the prior steps yielded no pools, check it against the Unicast Hash Table (PFUTA). If there is a match, add the pool for which their PFVML2FLT.ROPE (*Accept Unicast Hash*) bit is set. Refer to the software limitations described after [Step 8](#).
4. **VLAN groups** — This step is relevant only when VLAN filtering is enabled by the VLNCTRL.VFE bit.
 - a. Tagged packets: enable only pools in the packet's VLAN group as defined by the VLAN filters — PFVLVF[n] and their pool list — PFVLVFB[n] or pools for which the PFVML2FLT.VPE (VLAN Promiscuous Enable) is set.
 - b. Untagged packets: enable only pools with their PFVML2FLT.AUPE bit set.
 - c. If there is no match, the pool list should be empty.

5. **Default Pool** — If the pool list is empty at this stage and the PFVTCTL.DIS_DEF_POOL bit is cleared, set the default pool bit in the target pool list (from PFVTCTL.DEF_PL).
6. **Multicast or Broadcast** — If the packet is a multicast or broadcast packet with no external tag and was not forwarded in [Step 1](#) and [Step 2](#), set the default pool bit in the pool list (from PFVTCTL.DEF_PL).
7. **EtherType filters** — If one of the EtherType filters (ETQF) is matched by the packet and queuing action is requested and the *Pool Enable* bit in the ETQF is set, the pool list is set to the pool pointed by the filter.
8. **PFVFRE** — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list. The PFVFRE register blocks reception by a VF while the PF configures its registers.

The following software limitations apply when replication is disabled:

- Software must not set more than one bit in the bitmaps of the exact filters. Note that multiple bits can be set in an RAH register as long as it's guaranteed that the packet is sent to only one queue by other means (such as VLAN).
- Software must not set per-VM promiscuous bits (multicast or broadcast).
- Software must not set the *ROPE* bit in more than one PFVML2FLT register.
- Software should not activate mirroring.
- Software should not filter out local packets by setting PFFLP bits

7.8.3.4 Tx Packets Switching

Tx switching is used only in a virtualized environment to serve VM-to-VM traffic. Packets that are destined to one or more local VMs are directed back (loopback) to the Rx packet buffers. Enabling Tx switching is done by setting the PFDTXGSWC.LBE bit. Tx to Rx switching always avoids packet drop as if flow control is enabled. Therefore, the software must set the FCRT[n].RTH fields regardless if flow control is activated on the E610.

Tx switching rules are very similar to Rx switching in a virtualized environment, with the following exceptions:

- If a target pool is not found, the default pool is used only for broadcast and multicast packets.
- A unicast packet that matches an exact filter is not sent to the LAN.
- Broadcast and multicast packets are always sent to the external LAN.
- A packet might not be sent back to the originating pool (even if the destination address is equal to the source address) unless loopback is enabled for that pool by the PFVMTXSW[n] register.

The detailed flow for pool selection as well as the rules that apply to loopback traffic is as follows:

- Loopback is disabled when the network link is disconnected. It is expected (but not required) that system software (including VMs) does not post packets for transmission when the link is disconnected.
- Loopback is disabled when the *RXEN (Receive Enable)* bit is cleared.
- Loopback packets are identified by the *LB* bit in the receive descriptor.

Notes: When Tx switching is enabled, the host must avoid sending packets longer than 9.5 KB.
Tx Switching should be enabled only if PFVTCTL.POOLING_MODE = 00b (MAC mode).

7.8.3.4.1 Replication Mode Enabled

When replication mode is enabled, the pool list for any packet is determined according to the following steps:

1. **Exact unicast or multicast match** — If there is a match in one of the exact filters (RAL/RAH), for unicast or multicast packets, take the MPSAR[n] bits as a candidate for the pool list. Note that MPSAR[n] must not enable more than one pool for unicast RAL/RAH filters.
2. **EtherType filters** — If one of the enabled (ETQF.TX_ANTISPOOF) Transmit EtherType filters (ETQF.ETYPE) is matched by the packet and the appropriate Pool loopback/anti-spoof bit in the PFVFSPOOF is set (PFVFSPOOF.ETHERTYPELB/PFVFSPOOF.ETHERTYPEAS), the pool list is set to the pool pointed to by the ETQS matching register (loopback) or the packet is dropped (anti-spoof). All the subsequent steps after [Step 3](#) are skipped.

Note: Loopback on ETQF match is possible even if the general loopback enable bit (PFDTXGSWC.LBE) is clear.
3. **PFVFRE** — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list.
4. **Broadcast** — If the packet is a broadcast packet, add pools for which their PFVML2FLT.BAM bit (*Broadcast Accept Mode*) is set.
5. **Unicast hash** — If the packet is a unicast packet, and the prior steps yielded no pools, check it against the Unicast Hash Table (PFUTA). If there is a match, add pools for which their PFVML2FLT.ROPE bit (*Accept Unicast Hash*) is set.
6. **Multicast hash** — If the packet is a multicast packet and the prior steps yielded no pools, check it against the Multicast Hash Table (MTA). If there is a match, add pools for which their PFVML2FLT.ROMPE bit (*Receive Multicast Packet Enable*) is set.
7. **Unicast Promiscuous** — If the packet is a unicast packet, take the candidate list from prior steps and add pools for which their PFVML2FLT.UPE bit (*Unicast Promiscuous Enable*) is set.
8. **Multicast Promiscuous** — If the packet is a multicast packet, take the candidate list from prior steps and add pools for which their PFVML2FLT.MPE bit (*Multicast Promiscuous Enable*) is set.
9. **Filter source pool** — The pool from which the packet was sent is removed from the pool list unless the PFVMTXSW.LLE bit is set.
10. **VLAN groups** — This step is relevant only when VLAN filtering is enabled by the VLNCTRL.VFE bit.
 - a. Tagged packets: enable only pools in the packet's VLAN group as defined by the VLAN filters — PFVLVF[n] and their pool list — PFVLVFB[n] or pools for which the PFVML2FLT.VPE (*VLAN Promiscuous Enable*) is set.
 - b. Untagged packets: enable only pools with their PFVML2FLT.AUPE bit set.
 - c. If there is no match, the pool list should be empty.
11. **Forwarding to the network** — Packets are forwarded to the network in the following cases:
 - a. All broadcast and multicast packets.
 - b. Unicast packets that do not match any exact filter. A match of an exact filter that also points to a pool disabled via PFVFRE is not considered a match.
12. **PFVFRE** — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list (pre mirroring step).

13. **Mirroring** — Each of the following three mirroring rules adds its destination pool (PFMRCTL.MP) to the pool list if the following applies:
 - a. **Pool mirroring** — PFMRCTL.VPME is set and one of the bits in the pool list matches one of the bits in the PFMRVM register.
 - b. **VLAN port mirroring** — PFMRCTL.VLME is set and the index of the VLAN of the packet in the PFVLVF table matches one of the bits in the PFMRVLAN register.
 - c. **Downlink port mirroring** — PFMRCTL.DPME is set and the packet is sent to the network.
14. **PFVFRE** — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list (post mirroring step).

7.8.3.4.2 Replication Mode Disabled

When replication mode is disabled, software should take care of multicast and broadcast packets and check which of the VMs should get them. In this mode the pool list for any packet always contains one pool only according to the following steps:

1. **Exact unicast or multicast match** — If the packet DA matches one of the exact filters (RAL/RAH), take the MPSAR[n] bits as a candidate for the pool list. Note that MPSAR[n] must not enable more than one pool for unicast RAL/RAH filters.
2. **EtherType filters** - if one of the enabled (ETQF.TX_ANTISPOOF) Transmit EtherType filters (ETQF.ETYPE) is matched by the packet and the appropriate Poll loopback/anti-spoof bit in the PFVFSPOOF is set (PFVFSPOOF.ETHERTYPELB/PFVFSPOOF.ETHERTYPEAS), the pool list is set to the pool pointed to by the ETQS matching register (loopback) or the packet is dropped (anti-spoof). All the subsequent steps after [Step 3](#) are skipped.

Note: Loopback on EtherType match is possible even if the general loopback enable bit (PFDTXGSWC.LBE) is clear.

3. **PFVFRE** — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list.
4. **Unicast hash** — If the packet is a unicast packet, and the prior steps yielded no pools, check it against the Unicast Hash Table (PFUTA). If there is a match, add the pool for which their PFVML2FLT.ROPE bit (*Accept Unicast Hash*) is set. Refer to the software limitations that follow.
5. **VLAN groups** — This step is relevant only when VLAN filtering is enabled by the VLNCTRL.VFE bit.
 - a. Tagged packets: enable only pools in the packet's VLAN group as defined by the VLAN filters — PFVLVF[n] and their pool list — PFVLVFB[n] or pools for which the PFVML2FLT.VPE (*VLAN Promiscuous Enable*) is set.
 - b. Untagged packets: enable only pools with their PFVML2FLT.AUPE bit set.
 - c. If there is no match, the pool list should be empty.
6. **Multicast or Broadcast** — If the packet is a multicast or broadcast packet and was not forwarded in [Step 1](#) and [Step 2](#), set the default pool bit in the pool list (from PFVTCTL.DEF_PL).
7. **Filter source pool** — The pool from which the packet was sent is removed from the pool list unless the PFVMTXSW.LLE bit is set.
8. **Forwarding to the Network** — Packets are forwarded to the network in the following cases:
 - a. All broadcast and multicast packets.
 - b. Unicast packets that do not match any exact filter. A match of an exact filter that also points to a pool disabled via PFVFRE is not considered a match.
9. **PFVFRE** — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list.

The following software limitations apply when replication is disabled:

1. It is software's responsibility not to set more than one bit in the bitmaps of the exact filters. Note that multiple bits can be set in an RAH register as long as it is guaranteed that the packet is sent to only one queue by other means (such as VLAN)
2. Software must not set per-VM promiscuous bits (multicast or broadcast).
3. Software must not set the *ROPE* bit in more than one PFVML2FLT register.
4. Software should not activate mirroring.

7.8.3.5 Mirroring Support

The E610 supports four separate mirroring rules, each associated with a destination pool (mirroring can be done in up to four pools). Each rule is programmed with one of the four mirroring types:

- **Pool Mirroring** — Reflects all the packets received to a pool from the network.
- **Uplink Port Mirroring** — Reflects all the traffic received from the network that reached any pool.
- **Downlink Port Mirroring** — Reflects all the traffic transmitted to the network.
- **VLAN Mirroring** — Reflects all the traffic received from the network in a set of given VLANs (either from the network or from local VMs).

Note: Reflecting all the traffic received by any of the pools (either from the network or from local VMs) is supported by enabling mirroring of all pools.

Note: To get all traffic irrespective of the filtering rules, the mirroring pool should be set to promiscuous mode and promiscuous VLAN mode.

Mirroring modes are controlled by a set of rule control registers:

- **PFMRCTL** — Controls the rules to be applied and the destination port.
- **PFMRVLAN** — Controls the VLAN ports as listed in the PFVLVF table taking part in the VLAN mirror rule.
- **PFMRVM** — Controls the pools taking part in the pool mirror rule.

7.8.3.6 Offloads

The general rule is that offloads are executed as configured for the pool and queue associated with the receive packet. Some special cases:

- If a packet is directed to a single pool, offloads are determined by the pool and queue for that packet.
- If a packet is replicated to more than one pool, each copy of the packet is offloaded according to the configuration of its pool and queue.
- If replication is disabled, offloads are determined by the unique destination of the packet.

The following subsections describe exceptions to the previously described special cases.

7.8.3.6.1 Local Traffic Offload

The following capabilities are not supported on the loopback path:

- The RSS and VLAN strip offload capabilities are only supported if the *CC* bit in the transmit descriptor of the packet is set and if it is a tunnel packet, the *TUNNEL. OUTERIPCS* is set also. The reason is that when these bits are not set, software does not provide the necessary offload offsets with the Tx packet.

7.8.3.6.2 Rx Traffic Offload

- CRC offload is a global policy. CRC strip is enabled or disabled for all received packets.

7.8.3.7 Rate Control Features

7.8.3.7.1 Congestion Control

Tx packets going through the local switch are stored in the Rx packet buffer, similar to packets received from the network. Tx to Rx switching always avoids packet drop as if flow control is enabled. Therefore, the software must set the *FCRTH[n].RTH* fields regardless if flow control is activated on the E610.

The E610 guarantees that one TC flow is not affected by congestion in another TC.

Receive and local traffic are provided with the same priority and performance expectations. Packets from the two sources are merged in the Rx packet buffers, which can in general support both streams at full bandwidth. Any congestion further in the pipeline (such as lack of PCIe bandwidth) evenly affects Rx and local traffic.

7.8.3.7.2 Tx Queue Arbitration and Rate Control

To guarantee each pool with adequate bandwidth, a per-pool bandwidth control mechanism is added to the E610. Each Tx pool gets a percentage of the transmit bandwidth and is guaranteed it can transmit within its allocation. This arbitration is combined with the TC arbitration. See additional details on DCB Tx capabilities in [Section 7.6.2](#).

7.8.3.7.3 Receive Priority

As the switch might decide to loopback packets from the transmit path to the receive path, in case the receive path is full, the transmit path might be blocked (including the traffic to the LAN). The E610 guarantees that packets are not dropped because of that and that one traffic class flow is not affected by congestion in another traffic class. The PF driver might decide to program the E610 to drop packets from receive queues without available descriptors.

To keep the congestion effect locality, receive traffic from the LAN have higher priority than loop back traffic. This way large loopback traffic does not impact the network.

7.8.3.8 Small Packets Padding

In Virtualized systems, the driver receiving the packet in the VM might not be aware of all the hardware offloads applied to the packet. Thus, in case of stripping actions by the hardware (VLAN strip), it might receive packets which are smaller than a legal packet. The E610 provides an option to pad small packets in such cases so that all packets have a legal size. This option can be enabled only if the CRC is stripped. In these cases, all small packets are padded to 60 bytes (legal packet - 4 bytes CRC). The padding is done with zero data. This function is enabled via the `RDRXCTL.PSP` bit.

Packet padding is done after tag extractions and do not take into account if tags are exposed in the Rx descriptor status.

Notes:

- The packet length reported in the descriptor reflects the packet length before padding.
- If a timestamp is added, it is part of the padding data.

7.8.3.9 Switch Control

The PF driver has some control of the switch logic. The following registers are available to the PF for this purpose:

- **PFVTCTL (PF Virtual Control Register)**
 - **RPL_EN (Replication Enable)** — Enables replication of multicast and broadcast packets — both in incoming and local traffic. If this bit is cleared, Tx multicast and broadcast packets are sent only to the network and Rx multicast and broadcast packets are sent to the default pool.
 - **DEF_PL (Default Pool)** — Defines the target pool for packets that passed L2 filtering but did not pass any of the pool filters. This field is invalid when the `DIS_DEF_POOL` bit is set.
 - **DIS_DEF_POOL (Disable Default Pool)** — Disables acceptance of packets that failed all pool filters.
- **PFVFRE (PF VF Receive Enable)** — Enables/disables reception of packets from the link to a specific VF. Used during initialization of the VF. See [Section 4.2.2.2](#) for more details.
- **PFDTXGSWC (PF DMA Tx General Switch Control)**
 - **LBE (Loopback Enable)** — VMDQ loopback enables switching of Tx traffic to the Rx path for VM-to-VM communication.
- **PFVFSPOOF (PFVF Anti-Spoof Control)**
 - **MACAS (MAC Anti-Spoof Enable)** — Enables filtering of Tx packet for anti-spoof.
- **PFVMTXSW (PF VM Tx Switch Loopback Enable)**
 - **LLE (Local Loopback Enable)** — Local Loopback Enable defines whether to allow loopback of a packet from a certain pool into itself.
- **PFQDE (PF Queue Drop Enable Register)** — A register defining global policy for drop enable functionality when no descriptors are available. It lets the PF override the per-queue `SRCTL[n] DROP_EN` setting. PFQDE should be used in SR-IOV mode as described in [Section 4.6.10.3.1](#).
- **PFVML2FLT (PF VM L2Control Register)**
 - **ROMPE (Receive Overflow Multicast Packets)** — Accept multicast hash. Defines whether a pool accepts packets that match the multicast MTA table.

- **ROPE (Receive MAC Filters Overflow)** — Accept unicast hash. Defines whether a pool accepts packets that match the unicast PFUTA table.
- **BAM (Broadcast Accept)** — Defines whether a pool accepts broadcast packets.
- **MPE (Multicast Promiscuous Enable)** — Defines whether or pool accepts all multicast packets.
- **AUPE (Accept Untagged Packets Enable)** — Defines whether a pool accepts untagged VLAN packets.
- **Mirror Control** — See [Section 7.8.3.5](#).
- **PFVFTE (PF VF Transmit Enable)** — Enables/disables transmission of packets to the link to a specific VF. Used during initialization of the VF. See [Section 4.2.2.2](#) for more details.
- **PFVLVF/PFVLVFB (PF VM VLAN Pool Filter/Bitmap)** — VLAN queuing table. A set of 64 VLAN entries with an associated bitmap, one bit per pool. Bits are set for each pool that participates in this VLAN.
- **PFUTA (PF Unicast Table Array)** — A 4 Kb array that covers all combinations of 12 bits from the MAC destination address. A received unicast packet that misses the MAC filters is compared against the PFUTA. If the relevant bit in the PFUTA is set, the packet is routed to all pools for which the *ROPE* bit is set.
- **MTA (Multicast Table Array)** — A 4 Kb array that covers all combinations of 12 bits from the MAC destination address. A received multicast packet that misses the MAC filters is compared against the MTA. If the relevant bit in the MTA is set, the packet is routed to all pools for which the *ROMPE* bit is set.

In addition, the rate-control mechanism is programmed as described in [Section 7.6.2.1](#).

7.8.4 Security Features

The E610 allows some security checks on the inbound and outbound traffic of the switch.

7.8.4.1 Inbound Security

Each incoming packet (either from the LAN or from a local VM) is filtered according to the VLAN tag so that packets from one VLAN cannot be received by pools that are not members of that VLAN.

7.8.4.2 Outbound Security

7.8.4.2.1 MAC Anti-Spoofing

Each pool is associated with one or more Ethernet MAC Addresses on the receive path. The association is determined through the MPSAR registers. The MAC anti-spoofing capability insures that a VM always uses a source Ethernet MAC Address on the transmit path that is part of the set of Ethernet MAC Addresses defined on the Rx path. A packet with a non-matching SA is dropped, preventing spoofing of the Ethernet MAC Address. This feature is enabled in the *PFVFSPOOF.MACAS* field, and can be enabled per Tx pool.

Note: Anti-spoofing is not available for VMs that hide behind other VMs whose Ethernet MAC Addresses are not part of the RAH/RAL Ethernet MAC Address registers. In this case, anti-spoofing should be done by software switching, handling these VMs.

7.8.4.2.2 VLAN Anti-Spoofing

Each pool is associated with one or more VLAN tags on the receive path. The association is determined through the PFVLVF and PFVLVFB registers. The VLAN anti-spoofing capability insures that a VM always uses a VLAN tag on the transmit path that is part of the set of VLAN tags defined on the Rx path. A packet with a non-matching VLAN tag is dropped, preventing spoofing of the VLAN tag. This feature is enabled in the PFVFSPOOF.VLANAS field, and can be enabled per Tx pool.

Notes: If VLAN anti-spoofing is enabled, MAC anti-spoofing must be enabled as well.

When double VLAN is enabled by DMATXCTL.GDV and it is expected to transmit untagged packets by software, VLAN anti-spoofing should not be enabled.

Un-tagged packets are not checked by VLAN anti-spoofing.

7.8.4.2.3 VLAN Tag Validation

In PCI-SIG IOV scenarios the driver might be malicious, and thus might fake a VLAN tag. The E610 provides the ability to force a specific VLAN value for a VM. The possible behaviors are controlled by the PFVMVIR[n] registers as follows:

- **Use descriptor value** — To be used in case of a trusted VM that can decide which VLAN to send. This option should also be used in case one VM is member of multiple VLANs.
- **Always insert default VLAN** — This mode should be used for non-trusted or non-VLAN aware VMs. In this case, any VLAN insertion command from the VM is ignored. If a packet is received from the host with a VLAN, the packet should be dropped.
- **Never insert VLAN** — This mode should be used in a non-VLAN network. In this case, any VLAN insertion command from the VM is ignored. If a packet is received with a VLAN, the packet should be dropped.

Notes: The VLAN insertion settings should be done before any of the queues of the VM are enabled.

When double VLAN is enabled by DMATXCTL.GDV and it is expected to transmit untagged packets by software, VLAN validation should not be enabled.

7.8.4.2.4 E-tag Insertion

In addition to the VLAN insertion described above, the device can add an E-tag to the packets sent by a VF. The PFVMVIR[n].TAGA bit defines if a tag should be added. If this bit is set, the added tag is as follows:

- TAGA = 00b (MAC mode) or 11b (Reserved): Insert no tag.
- TAGA = 01b (E-tag mode): An E-tag is added as follows:

{TAG_ETYPE.ETAG_ETHERTYPE[15:0], PFVMTIR.PORT_TAG_ID[31:0], 0x0000}

Notes: The E-tag insertion settings should be done before any of the queues of the VM are enabled.

7.8.4.2.5 EtherType Anti-Spoofing/Pruning

Each pool can be set to drop packets associated with one or more (up to 8) EtherTypes on the transmit path. The EtherTypes are defined using the ETQF registers. The EtherType pruning (anti-spoof) capability ensures that a VM does not send out LLDP/ECP control frames to external switches. The EtherType loopback capability enables the loopback of link layer packets to a pre defined pool destination.

EtherType pruning (anti-spoof) feature is enabled in the `PFVFSPOOF.ETHERTYPEAS` field, and can be enabled per transmit pool. EtherType loopback feature is enabled in the `PFVFSPOOF.ETHERTYPELB` field, and can be enabled per transmit pool.

Note: If one of the ETQF filters was set to check EtherType `TX_ANTISPOOF`, then the `ETHERTYPELB` and `ETHERTYPEAS` fields must be set as well.

7.8.4.3 Malicious Driver Detection

The hardware can be programmed to take some action as a result of some misbehavior of a VM. These actions might hint to the fact that some VM is malicious and the VMM should remedy the situation. To inform the VMM of this fact, The Mailbox bit (`EICR.MAILBOX` bit) may be set to indicate the occurrence of such behavior. The `LMVM_RX` and `LMVM_TX` register contains information on which queue (`LMVM_TX.LAST_Q` or `LMVM_RX.LAST_Q`) and port (`LMVM_TX.MAL_PF` and `LMVM_RX.MAL_PF`) the malicious behavior was detected. The `LVMMC_TX` and `LVMMC_RX` registers contain information on the type of error detected and are clear by read.

Malicious driver behavior detection is enabled by setting the `DMATXCTL.MDP_EN` and `RDRXCTL.MDP_EN` bits to 1. This capability should be enabled before queues are exposed to non trusted virtual machines.

On detection of a malicious driver event the E610 stops activity of the offending queue, and sets the bit matching the offending queue in `WQBR_RX` and `WQBR_TX` registers. If `DMATXCTL.MBINTEN` or `RDRXCTL.MBINTEN` are set, the E610 generates an interrupt for transmit or receive errors respectively by asserting the `EICR.MAILBOX` bit. Cause of Malicious driver activation is reported in the `LVMMC_TX` and `LVMMC_RX` registers. To re-activate offending queue, driver should release it by setting the matching bit in `WQBR_RX` or `WQBR_TX` register.

The `WQBR_RX` and `WQBR_TX` registers keeps track of all the queues on which a malicious activity was detected and can be used by the driver to track multiple events.

After the queue is disabled the initialization flow defined in [Section 4.6.7.1](#) for receive queues and in [Section 4.6.8.1](#) for transmit queues should be applied.

7.8.4.3.1 Queue Context Validation

The E610 checks that the queue context submitted when a queue is enabled is valid. It also prevents change to static configuration while the queue is enabled. These checks are done both for receive and transmit queues. The table below describes the checks done on the queue context:

Check type	Description	Bit in LVMMC_TX/RX	Action
Receive queue context validity	Check that: - RDLEN > 0 - SRRCTL.BSIZEPACKET > 0 - SRRCTL.BSIZEHEADER > 0 if DESCTYP requires header split - SRRCTL.BSIZEHEADER <= 1024 - SRRCTL.DESCTYPE is valid	INVALID_RXQ_CONTEXT	Prevent Enable of queue
Receive queue context change on the fly	Attempt to write RDBAL, RDBAH, RDLEN, or SRRCTL while queue is enabled		Disable queue.
Transmit queue context validity	Check that: TDLEN > 0	INVALID_TXQ_CONTEXT	Prevent Enable of queue
Transmit queue context change on the fly	Attempt to write TDBAL, TDBAH, TDLEN, TDWBAL, or TDWBAH while queue is enabled		Disable queue.

Note: Queue zero is enabled by default. Thus, if malicious driver protection is enabled, to change Queue zero configuration, the queue should be disabled.

7.8.4.3.2 Transmit Descriptor Validity Checks

The table below describes the checks are done by the E610 to define if a transmit packet descriptor is valid. All the checks are done on the descriptors. The checks on the packet header are described in the previous sections.

Table 7-69. Malicious Driver - TX Descriptor Checks

Check type	Description	Bit in LVMMC_TX	Action
Mac Header size	Checks that the MAC header size in the context descriptor is at least 14 (or 18 in case of offloaded packet and double VLAN).	MAC_HEADER	Drop Packet and stop offending queue
IPv4 Header	If a checksum or TSO offload is required, checks that the IPv4 header size in the context descriptor is at least 20.	IPv4_HEADER	Drop Packet and stop offending queue
IPv6 Header	If a TSO offload is required, checks that the IPv6 header size in the context descriptor is at least 40.	IPv6_HEADER	Drop Packet and stop offending queue
Wrong MAC_IP	Check that the MAC+ IP (+ OUTERIP + TUNNEL) header size is not bigger than the packet size.	WRONG_MAC_IP	Drop Packet and stop offending queue
TCP header size	If a TCP TSO offload is required, checks that the TCP header size in the context descriptor is at least 20.	TCP_LSO	Drop Packet and stop offending queue
UDP header size	If a UDP TSO offload is required, checks that the TCP header size in the context descriptor is at least 8.	UDP_LSO	Drop Packet and stop offending queue
SCTP data size	If a SCTP checksum offload is required, checks that the SCTP L4 packet size (including header and data) is at least 12.	STCP_CS	Drop Packet and stop offending queue
Packet too big	In case of a single send, check that the packet is not larger than 15.5 KBytes in normal mode, and not larger than 10300 Bytes in TX packet switching mode.	SIZE	Drop Packet and stop offending queue
Packet too small	Check that the total length of the packet transmitted, not including FCS is at least 13 bytes.	N/A	Silently drop packet.
Illegal offload request.	Check that TSO is no requested for SCTP or that a checksum offload is not requested for IPv6 packets.	OFF_ILL	Drop Packet and stop offending queue
SCTP alignment	If an SCTP CRC offload is requested, check that the data size is 4 byte aligned.	SCTP_ALIGNED	Drop Packet and stop offending queue
Zero MSS	Check that the MSS size is larger than zero.	ZERO_MSS	Drop Packet and stop offending queue
Context in middle of packet	Check that a context descriptor is not sent in the middle of a packet.	CONTEXT_IN_PACKET	Drop Packet and stop offending queue
Number of large send header buffers	Check that the Large send header is contained in at most 4 buffers.	LSO_MORE_THAN_4	Drop Packet and stop offending queue
Buffers size and length match - Single Send	For single send, check that the total of all buffers size and the packet length match.	OOS_SSO	Drop Packet and stop offending queue
Buffers size and length match - Large Send	For LSO, check that the total of all buffers size and the packet length match.	OOS_LSO	Drop Packet and stop offending queue
LSO wrong header size	For LSO, check that the size of the header fetched actually match the expected header size.	N/A	Silently drop packet. Counted in SSVPC
UDP data size	If a UDP checksum offload is required, checks that the UDP L4 packet size in the context descriptor is at least 8.	SSO_UDP	Drop Packet and stop offending queue

Table 7-69. Malicious Driver - TX Descriptor Checks [continued]

Check type	Description	Bit in LVMMC_TX	Action
TCP data size	If a TCP checksum offload is required, checks that the TCP L4 packet size in the context descriptor is at least 20.	SSO_TCP	Drop Packet and stop offending queue
Descriptor Type	Check that only descriptor types 2 (context) or 3 (advanced data descriptor) are used.	DESC_TYPE	Drop Packet and stop offending queue
CC bit not set when needed	CC (Check Context) bit in descriptor is not set in virtualization mode.		Drop Packet and stop offending queue
Null packet check	Check that a Null packet has the EOP bit set.	WRONG_NULL	Drop Packet and stop offending queue
Packet without EOP	Check that a only entire packets are provided by the driver	NO_EOP	Drop Packet and stop offending queue
Burst of contexts	Check that less than 4 Contiguous context descriptor are sent by the driver.	CONTEXT_BURST	Drop Packet and stop offending queue
Legacy Descriptor in SR-IOV	Legacy descriptor used when SR-IOV is enabled.	LEGACY_DESC_IOV	Drop Packet and stop offending queue

7.8.4.3.3 Reactive Malicious Behavior Detection

The table below describes the checks are done by the E610 to detect a malicious behavior, even if the packet seems valid.

Table 7-70. Reactive Malicious Checks

Check type	Description	Bit in LVMMC_TX/RX	Action
Malicious VF memory access	A PCIe DMA access initiated by a VF ended with Unsupported Request (UR) or Completer Abort (CA). This check is done for both Tx and Rx queues.	INV_MACC	Drop Packet and stop offending queue
DMA access outside of active memory ¹	A Queue attempt to read memory outside of the active memory range of the system. See Section 3.1.5.7 for details.	INV_MACC	Drop PCIe read transaction. Stop queue that requested the transaction
	A Queue attempt to write memory outside of the active memory range of the system. See Section 3.1.5.7 for details.	N/A	Silently drop PCIe write transaction.
VLAN not expected in packet	A packet where VLAN should be added by device is in Tx descriptor. See Section 7.8.4.2.3	VLAN_IERR	Drop Packet
Anti-spoof checks	See Section 7.8.4.2.1 and Section 7.8.4.2.2 and Section 7.8.4.2.5 for details	MAC_VLAN_SPOOF	Drop Packets.

1. A PCIe error interrupt can be asserted when such a transaction is dropped. See [Section 3.1.5](#) for details.

7.8.5 Virtualization of Hardware

This section describes additional features used in both IOV and VMDq2 modes.

7.8.5.1 Per-Pool Statistics

Part of the statistics are by definition shared and cannot be allocated to a specific VM. For example, CRC error count cannot be allocated to a specific VM, as the destination of such a packet is not known if the CRC is wrong.

All the non-specific statistics are handled by the PF driver in the same way it is done in non-virtualized systems. A VM might request a statistic from the PF driver but might not access it directly.

The conceptual model used to gather statistics in a virtualization context is that each queue pool is considered as a virtual link and the Ethernet link is considered as the uplink of the switch. Thus, any packet sent by a pool is counted in the Tx statistics, even if it was forwarded to another pool internally or was dropped by the MAC for some reason. In the same way, a replicated packet is counted in each of the pools receiving it.

The following statistics are provided per pool:

- Good packet received count
- Good packet transmitted count
- Good octets received count
- Good octets transmitted count
- Multicast packets received count

Note: All the per VF statistics are read only and wrap around after reaching their maximum value.

7.9 Tunneling Support

The E610 supports the VXLAN and NVGRE tunneling packet formats. As part of this support the following features are supported:

- LSO and transmit checksum offloads for tunneled packets as described in [Section 7.2.4](#) and [Section 7.2.5](#).
- RSS forwarding based on inner L3/L4 header as described in [Section 7.1.3.6](#)
- Flow director forwarding based on Tenant ID, inner MAC and inner VLAN as described in [Section 7.1.3.5](#)
- New indications in the receive descriptor with information on the tunnel headers and on the outer header checksum as described in [Section 7.1.5.2](#)
- New packet split modes based on the tunnel header or outer L2 header as described in [Section A.2.5](#)

7.10 Reliability

7.10.1 Memory Integrity Protection

All E610 internal memories are protected against soft errors. Most of them are covered by ECC that correct single error per memory line and detect double errors per memory line. Few of the smaller memories are covered by parity protection that detects a single error per memory line.

Single errors in memories with ECC protection are named also as correctable errors. Such errors are silently corrected. Two errors in memories with ECC protection or single error in memories with parity protection are also named as un-correctable errors. Un-correctable errors are considered as fatal errors. If an un-correctable error is detected in Tx packet data, the packet is transmitted with a CRC error. If un-correctable error is detected in Rx packet data, the packet is reported to the host (or manageability) with a CRC error. If an un-correctable error is detected anywhere else, the E610 halts the traffic and sets the ECC error interrupt. Software is then required to initiate a complete initialization cycle to resume normal operation.

7.10.2 PCIe Error Handling

For PCIe error events and error reporting see [Section 3.1.5](#).

NOTE: *This page intentionally left blank.*

Chapter 8 Programming Interface

8.1 General

The E610's address space is mapped into four regions with PCI Base Address Registers listed in [Table 8-1](#) and explained more in [Section 9.2.2.16](#) and [Section 9.2.2.17](#).

Table 8-1. E610 Address Regions

Addressable Content	Mapping Style	Region Size
Internal registers memories and NVM (Memory BAR)	Direct memory mapped	1 MB + NVM Size ¹
FLASH (optional) ²	Direct memory mapped	2 MB - 16MB
Expansion ROM (optional)	Direct memory mapped	64 KB - 512 KB
Internal registers and memories (optional) ³	I/O window mapped	32 bytes
MSI-X (optional)	Direct memory mapped	16 KB ⁴

1. The Flash size is defined by the BARCTRL register.
2. The Flash space in the Memory CSR and Expansion ROM Base Address are mapped to different Flash memory regions. Accesses to the Memory BAR at offset 1 MB are mapped to the Flash device at offset 0x0, while accesses to the Expansion ROM at offset 0x0 are mapped to the Flash region pointed by the PXE Driver Module Pointer (read from NVM word address 0x05). See [Section 6.1](#). The Expansion ROM region has a size limited to 512 KB.
3. The internal registers and memories can be accessed through I/O space as explained in the sections that follow.
4. See [Section 9.2.2.16](#) for the MSI-X BAR offset in 32-bit and 64-bit BAR options.

8.1.1 Memory-Mapped Access

8.1.1.1 Memory-Mapped Access to Internal Registers and Memories

The internal registers and memories can be accessed as direct memory-mapped offsets from the Memory CSR BAR. See the following sections for detailed description of the Device registers.

In IOV mode, this area is partially duplicated per VF. All replications contain only the subset of the register set that is available for VF programming.

8.1.1.2 Memory-Mapped Accesses to Flash

The external Flash can be accessed using direct memory-mapped offsets from the CSR base address register ().

8.1.1.3 Memory-Mapped Access to MSI-X Tables

The MSI-X tables can be accessed as direct memory-mapped offsets from the base address register (BAR3). The MSI-X registers are described in [Section 8.2.2.7, "MSI-X Table Registers"](#).

In IOV mode, this area is duplicated per VF.

8.1.1.4 Memory-Mapped Access to Expansion ROM

The option ROM module located in the external Flash can also be accessed as a memory-mapped expansion ROM. Accesses to offsets starting from the Expansion ROM Base address reference the Flash, provided that access is enabled through the NVM Initialization Control Word, and if the Expansion ROM Base Address register contains a valid (non-zero) base memory address.

8.1.2 I/O-Mapped Access

All internal registers and memories can be accessed using I/O operations. I/O accesses are supported only if an I/O Base Address is allocated and mapped (BAR2), the BAR contains a valid (non-zero value), and I/O address decoding is enabled in the PCIe configuration.

When an I/O BAR is mapped, the I/O address range allocated opens a 32-byte window in the system I/O address map. Within this window, two I/O addressable registers are implemented: IOADDR and IODATA. The IOADDR register is used to specify a reference to an internal register or memory, and then the IODATA register is used to access it at the address specified by IOADDR:

Table 8-2. I/O Map

Offset	Abbreviation	Name	RW	Size
0x00	IOADDR	Internal Register, Internal Memory, or Flash Location Address. 0x00000-0xFFFFF – Internal registers/memories	RW	4 bytes
0x04	IODATA	Data field for reads or writes to the internal register, internal memory, or Flash location as identified by the current value in IOADDR. All 32 bits of this register are read/write-able.	RW	4 bytes
0x08-0x1F	Reserved	Reserved.	O	4 bytes

8.1.2.1 IOADDR (I/O Offset 0x00; RW)

The IOADDR register must always be written as a DWord access. Writes that are less than 32 bits are ignored. Reads of any size returns a DWord of data. However, the chipset or CPU might only return a subset of that DWord.

For software programmers, the IN and OUT instructions must be used to cause I/O cycles to be used on the PCIe bus. Because writes must be to a 32-bit quantity, the source register of the OUT instruction must be EAX (the only 32-bit register supported by the OUT command). For reads, the IN instruction can have any size target register, but it is recommended that the 32-bit EAX register be used.

Bits 31 through 20 are ignored by the device and should be set to zero by the software. At hardware reset (LAN_PWR_GOOD) or PCI reset, this register value resets to 0x00000000. Once written, the value is retained until the next write or reset.

8.1.2.2 IODATA (I/O Offset 0x04; RW)

The IODATA register must always be written as a DWord access when the IOADDR register contains a value for the internal register and memories (such as 0x00000-0x1FFFC). In this case, writes that are less than 32 bits are ignored.

Writes and reads to IODATA when the IOADDR register value is in an undefined range (0x20000-0x7FFFC) should not be performed. Results cannot be determined.

Note: There are no special software timing requirements on accesses to IOADDR or IODATA. All accesses are immediate except when data is not readily available or acceptable. In this case, the E610 delays the results through normal bus methods (like split transaction or transaction retry).

Because a register/memory read or write takes two I/O cycles to complete, software must provide a guarantee that the two I/O cycles occur as an atomic operation. Otherwise, results can be non-deterministic from the software viewpoint.

8.1.2.3 Undefined I/O Offsets

I/O offsets 0x08 through 0x1F are considered to be reserved offsets with the I/O window. DWord reads from these addresses return 0xFFFF, while writes to these addresses are discarded.

8.1.3 Configuration Access to Internal Registers and Memories

To support legacy pre-boot 16-bit operating environments without requiring I/O address space, the E610 enables accessing CSRs via the configuration address space by mapping IOADDR and IODATA registers into the configuration address space. The register mapping in this case is listed in [Table 8-3](#).

Table 8-3. IOADDR and IODATA in Configuration Address Space

Configuration Address	Abbreviation	Name	RW	Size
0xF8	IOADDR	Internal register or internal memory location address. 0x00000-0xFFFFF – Internal registers and memories. 0x20000-0x7FFFFFFF – Undefined.	RW	4 bytes
0xFC	IODATA	Data field for reads or writes to the internal register or internal memory location as identified by the current value in IOADDR. All 32 bits of this register can be read from or written to.	RW	4 bytes

Software writes data to an internal CSR via the configuration space in the following manner:

1. CSR address is written to the IOADDR register where:
 - Bit 31 (IOADDR.*Configuration IO Access Enable*) of the IOADDR register should be set to 1b.
 - Bits 30:0 of IOADDR should hold the actual address of the internal register or memory being written to.
2. Data to be written is written into the IODATA register.
 - The IODATA register is used as a window to the register or memory address specified by IOADDR register. As a result, the data written to the IODATA register is written into the CSR pointed to by bits 30:0 of the IOADDR register.

3. IOADDR.*Configuration IO Access Enable* is cleared to avoid unintentional CSR read operations (that might cause clear by read) by other applications scanning the configuration space.

Software reads data from an internal CSR via the configuration space in the following manner:

1. CSR address is written to the IOADDR register where:
 - Bit 31 (IOADDR.*Configuration IO Access Enable*) of the IOADDR register should be set to 1b.
 - Bits 30:0 of IOADDR should hold the actual address of the internal register or memory being read.
2. CSR value is read from the IODATA register.
 - The IODATA register is used as a window to the register or memory address specified by the IOADDR register. As a result, the data read from the IODATA register is the data of the CSR pointed to by bits 30:0 of the IOADDR register.
3. IOADDR.*Configuration IO Access Enable* is cleared to avoid un-intentional CSR read operations (that might cause clear by read) by other applications scanning the configuration space.

Notes:

- When functioning in a D3 state, software should not attempt to access CSRs via the IOADDR and IODATA configuration registers.
- To enable CSR access via configuration space, software should set bit 31 to 1b (IOADDR.*Configuration IO Access Enable*) in the IOADDR register. Software should clear bit 31 of the IOADDR register after completing CSR access to avoid an unintentional clear-by-read operation by another application scanning the configuration address space.
- Bit 31 of the IOADDR register (IOADDR.*Configuration IO Access Enable*) has no effect when initiating access via I/O address space.
- Configuration access to 0x9C (IODATA) without setting bit 31 of the IOADDR register (IOADDR.*Configuration Access Enable*) must not result in an unsupported request.
- I/O-mapped access and CSR access via PCIe configuration space are mutually exclusive operating modes, and therefore PCIE_CNF.IO_SUP NVM bit must be cleared when the PCIE_CAPSUP.CSR_CONF_EN NVM bit is set, and vice versa.

8.1.4 Register Terminology

Type	Description
RC	Read Clear A register bit with this attribute is cleared after read. Writes have no effect on the bit value.
RO	Read Only If a register is read only, writes to this register have no effect.
ROS	Read Only Status Writes to ROS fields has no effect. The value of the field can change due to hardware events.
RSV	Reserved Field can return any value on read access and must be set to its initialized value on write access unless specified differently in the field description.
RW	Read/Write A register with this attribute can be read and written. If written since reset, the value read reflects the value written.
RW1C	Read/Write Clear A register with this attribute can be read and written. However, a write of a 1b clears (sets to 0b) the corresponding bit and a write of a 0b has no effect.
RW/RC	Read/Write and Read Clear
RWS	Read Write Set Register that is set to 1b by software by writing a 1b to the register, and cleared to 0b by hardware.
WO	Write Only Reading this register might not return a meaningful value.

8.1.5 VF Registers Allocated per Queue

Depending on configuration, each pool has 2, 4, or 8 queues allocated to it. Note that in IOV mode, any queues not allocated to a VF are allocated to the PF. The registers assigned to a queue are accessible both in its VF address space and in the PF address space. This section describes the address mapping of registers that belong to queues.

Section 7.8.2.7.2 defines the correspondence of queue indices between the PF and the VFs. For example, when in configuration for 32 VFs, queues 124-127 in the PF correspond to queues [3:0] of VF# 31.

Note: The queue pair of one VF should be assigned to the PF, therefore the maximum usable number of VFs is 63.

The queues are enumerated in each VF from 0 (i.e. [1:0], [3:0], or [7:0]). If a queue is allocated to a VF, its corresponding registers are accessible in the VF CSR space. Each register is allocated an address in the VF (relative to its base) according to its index in the VF space. Therefore, the registers of queue 0 in each VF are allocated the same addresses, which equal the addresses of the same registers for queue 0 in the PF. For example, RDH[0] in the VF space has the same relative address in each VF and in the PF (address 0x01010).

8.1.6 Non-Queue VF Registers

Registers that do not correspond to a specific queue are allocated addresses in the VF space according to these rules:

- Registers that are read-only by the VF (e.g. STATUS) have the same address in the VF space as in the PF space.
- Registers allocated per pool are accessed in the VF in the same location as pool [0] in the PF address space.
- Registers that are RW by the VF (e.g. CTRL) are replicated in the PF, one per VF, in adjacent addresses.

Note: Since the VF address space is limited to 16 KB, any register that resides above that address in the PF space cannot reside in the same address in the VF space and is therefore allocated in another location in the VF.

8.2 Device Registers - PF

8.2.1 BAR0 Registers Summary

Table 8-4. BAR0 Registers Summary

Offset / Alias Offset	Abbreviation	Name	Section
General Control Registers			
0x00000000	CTRL	Device Control Register	Section 8.2.2.1.1
0x00000008	STATUS	Device Status Register	Section 8.2.2.1.2
0x00000018	CTRL_EXT	Extended Device Control Register	Section 8.2.2.1.3
0x00000020	ESDP	Extended SDP Control	Section 8.2.2.1.4
0x00000030	MAC_GPIO	MAC GPIO Register	Section 8.2.2.1.5
0x00000200	LEDCTL	LED Control	Section 8.2.2.1.6
0x00000204	LEDSTAT	LED Status	Section 8.2.2.1.7
0x00005078	EXVET	Extended VLAN Ether Type - Receive	Section 8.2.2.1.8
0x00008224	EXVET_T	Extended VLAN Ether Type - Transmit	Section 8.2.2.1.9
0x00015F64	GRC	General Receive Control	Section 8.2.2.1.10
0x00015F58	I2CCMD	SFP I2C Command	Section 8.2.2.1.11
0x00015F5C	I2CPARAMS	SFP I2C Parameters	Section 8.2.2.1.12
0x00015FF0	DEV_FUNC_EN	Device and Functions Enable Control	Section 8.2.2.1.13
0x00015FEC	FACTPS	Function Active and Power State to Manageability	Section 8.2.2.1.14
NVM Registers			
0x00015FF8	EEC	EEPROM Mode Control Register	Section 8.2.2.2.1
0x00015F68	FLA	Legacy Flash Access Register	Section 8.2.2.2.2
Flow Control Registers			
0x00004294	MFLCN	MAC Flow Control Register	Section 8.2.2.3.1
0x0000431C	PFCTOP	Priority Flow Control Type Opcode	Section 8.2.2.3.2
0x00003200 + 0x4*n, n=0...3	FCTTVN[n]	Flow Control Transmit Timer Value n	Section 8.2.2.3.3
0x00003220 + 0x4*n, n=0...7	FCRTL[n]	Flow Control Receive Threshold Low	Section 8.2.2.3.4
0x00003260 + 0x4*n, n=0...7	FCRTH[n]	Flow Control Receive Threshold High	Section 8.2.2.3.5
0x000032A0	FCRTV	Flow Control Refresh Threshold Value	Section 8.2.2.3.6
0x0000CE00	TFCS	Transmit Flow Control Status	Section 8.2.2.3.7
0x00003D00	FCCFG	Flow Control Configuration	Section 8.2.2.3.8

Table 8-4. BAR0 Registers Summary [continued]

Offset / Alias Offset	Abbreviation	Name	Section
PCIe Registers			
0x00011028	PCI_STATUS1	PCIe Function Status 1	Section 8.2.2.4.1
0x00011050	GCR_EXT	PCIe Control Extended Register	Section 8.2.2.4.2
0x00011098	PCI_REVID	PCIe Revision ID	Section 8.2.2.4.3
0x00011540	PCI_DREVID	PCIe Default Revision ID	Section 8.2.2.4.4
0x00011070	FUNC_RID	Function Requester ID Information Register	Section 8.2.2.4.5
0x00011530	PCI_VMINDEX	PCIe VM Pending Index	Section 8.2.2.4.6
0x00011538	PCI_VMPEND	PCIe VM Pending Status	Section 8.2.2.4.7
0x00011528	PCI_IENA	PCIe Interrupts Enable	Section 8.2.2.4.8
0x00011520	PCI_ICAUSE	PCIe Interrupt Cause	Section 8.2.2.4.9
0x00011140	PCI_PCIERR	PCIe Errors Reported	Section 8.2.2.4.10
0x00011800	PCI_GSCL_1	PCIe Statistic Control Register #1	Section 8.2.2.4.11
0x00011804	PCI_GSCL_2	PCIe Statistic Control Registers #2	Section 8.2.2.4.12
0x00011820 + 0x4*n, n=0...3	PCI_GSCN_0_3[n]	PCIe Statistic Counter Registers #0...#3	Section 8.2.2.4.13
0x00011810 + 0x4*n, n=0...3	PCI_GSCL_5_8[n]	PCIe Statistic Control Register #5...#8	Section 8.2.2.4.14
0x00011548	PCI_BYTCTL	PCIe Byte Counter Low	Section 8.2.2.4.15
0x00011544	PCI_BYTCTH	PCIe Byte Counter High	Section 8.2.2.4.16
0x00011788	PCI_LCBADD	PCIe LCB Address Port	Section 8.2.2.4.17
0x00011734	PCI_LCBDATA	PCIe LCB Data Port	Section 8.2.2.4.18
0x00011040	PCI_FWCTRL	PCIe Firmware Control	Section 8.2.2.4.19
0x00011044	PCI_CSRT0	PCIe CSR Access Timeout	Section 8.2.2.4.20
0x00011054	PCI_FLASHTO	PCI Flash Access Timeout	Section 8.2.2.4.21
PCIe Configuration Space Setting Registers			
0x000110F8	PCI_CNF2	PCIe Global Config 2	Section 8.2.2.5.1
0x00011048	PCI_LBARCTRL	PCI BAR Control	Section 8.2.2.5.2
0x000110B8	PCI_GLBL_CNF	PCIe Global Config	Section 8.2.2.5.3
0x00011088	PCI_CAPSUP	PCIe Capabilities Support	Section 8.2.2.5.4
0x00011090	PCI_LINKCAP	PCIe Link Capabilities	Section 8.2.2.5.5
0x00011080	PCI_CAPCTRL	PCIe Capabilities Control	Section 8.2.2.5.6
0x000110F0	PCI_SERL	PCIe Serial Number MAC Address Low	Section 8.2.2.5.7
0x00011078	PCI_SERH	PCIe Serial Number MAC Address High	Section 8.2.2.5.8
0x00011000	PCI_CNF	PCIe PF Configuration	Section 8.2.2.5.9
0x00011038	PCI_CLASS	PCIe Storage Class	Section 8.2.2.5.10
0x000110A0	PCI_PMSUP	PCIe PM Support	Section 8.2.2.5.11

Table 8-4. BAR0 Registers Summary [continued]

Offset / Alias Offset	Abbreviation	Name	Section
0x00011060	PCI_PWRDATA	PCIe Power Data Register	Section 8.2.2.5.12
0x0001103C	PCI_VENDORID	PCIe Vendor ID	Section 8.2.2.5.13
0x00011008	PCI_PFDEVID	PCIe PF Device ID	Section 8.2.2.5.14
0x00011010	PCI_VFDEVID	PCIe VF Device ID	Section 8.2.2.5.15
0x000110E8	PCI_UPADD	PCIe Upper Address	Section 8.2.2.5.16
Interrupt Registers			
0x00000800	EICR	Extended Interrupt Cause Register	Section 8.2.2.6.1
0x00000808	EICS	Extended Interrupt Cause Set Register	Section 8.2.2.6.2
0x00000880	EIMS	Extended Interrupt Mask Set/Read Register	Section 8.2.2.6.3
0x00000888	EIMC	Extended Interrupt Mask Clear Register	Section 8.2.2.6.4
0x00000810	EIAC	Extended Interrupt Auto Clear Register	Section 8.2.2.6.5
0x00000890	EIAM	Extended Interrupt Auto Mask Enable register	Section 8.2.2.6.6
0x00000A90	EICS1	Extended Interrupt Cause Set Registers 1	Section 8.2.2.6.7
0x00000A94	EICS2	Extended Interrupt Cause Set Registers 2	Section 8.2.2.6.8
0x00000AA0	EIMS1	Extended Interrupt Mask Set/Read Registers 1	Section 8.2.2.6.9
0x00000AA4	EIMS2	Extended Interrupt Mask Set/Read Registers 2	Section 8.2.2.6.10
0x00000AB0	EIMC1	Extended Interrupt Mask Clear Registers 1	Section 8.2.2.6.11
0x00000AB4	EIMC2	Extended Interrupt Mask Clear Registers 2	Section 8.2.2.6.12
0x00000AD0	EIAM1	Extended Interrupt Auto Mask Enable registers 1	Section 8.2.2.6.13
0x00000AD4	EIAM2	Extended Interrupt Auto Mask Enable registers 2	Section 8.2.2.6.14
0x00000894	EITRSEL	MSIX to EITR Select	Section 8.2.2.6.15
0x00012300 + 0x4*n, n=0...104 and 0x00000820 + 0x4*(n-105), n=105...128	EITR[n]	Extended Interrupt Throttle Registers	Section 8.2.2.6.16
0x00000900 + 0x4*n, n=0...63	IVAR[n]	Interrupt Vector Allocation Registers	Section 8.2.2.6.17
0x00000A00	IVAR_MISC	Miscellaneous Interrupt Vector Allocation	Section 8.2.2.6.18
0x00000898	GPIE	General Purpose Interrupt Enable	Section 8.2.2.6.19

Table 8-4. BAR0 Registers Summary [continued]

Offset / Alias Offset	Abbreviation	Name	Section
MSI-X Table Registers			
0x000110C0 + 0x4*n, n=0...1	PBACL[n]	MSI-X PBA Clear	Section 8.2.2.7.1
0x000110C8 + 0x4*n, n=0...5	VFPBACL[n]	VF MSI-X PBA Clear	Section 8.2.2.7.2
Receive Registers			
0x00005080	FCTRL	Filter Control Register	Section 8.2.2.8.1
0x00005088	VLNCTRL	VLAN Control Register	Section 8.2.2.8.2
0x00005090	MCSTCTRL	Multicast Control Register	Section 8.2.2.8.3
0x0000EA00 + 0x4*n, n=0...63	PSRTYPE[n]	Packet Split Receive Type Register	Section 8.2.2.8.4
0x00005000	RXCSUM	Receive Checksum Control	Section 8.2.2.8.5
0x00005008	RFCTL	Receive Filter Control Register	Section 8.2.2.8.6
0x00005200 + 0x4*n, n=0...127	MTA[n]	Multicast Table Array	Section 8.2.2.8.7
0x0000A200 + 0x8*n, n=0...127	RAL[n]	Receive Address Low	Section 8.2.2.8.8
0x0000A204 + 0x8*n, n=0...127	RAH[n]	Receive Address High	Section 8.2.2.8.9
0x00005400 + 0x8*n, n=0...15	RAL_ALIAS[n]	Receive Address Low Alias	Section 8.2.2.8.10
0x00005404 + 0x8*n, n=0...15	RAH_ALIAS[n]	Receive Address High Alias	Section 8.2.2.8.11
0x0000A600 + 0x4*n, n=0...255	MPSAR[n]	MAC Pool Select Array	Section 8.2.2.8.12
0x0000A000 + 0x4*n, n=0...127	VFTA[n]	VLAN Filter Table Array	Section 8.2.2.8.13
0x0000EC80	MRQC	Multiple Receive Queues Command Register	Section 8.2.2.8.14
0x00005818	MRQC_ALIAS	Multiple Receive Queues Command Register Alias	Section 8.2.2.8.15
0x0000EC70	RQTC	RSS queues per Traffic class register	Section 8.2.2.8.16
0x0000EB80 + 0x4*n, n=0...9	RSSRK[n]	RSS Random Key Register	Section 8.2.2.8.17
0x0000EB00 + 0x4*n, n=0...31	RETA[n]	Redirection Table	Section 8.2.2.8.18
0x0000EE80 + 0x4*n, n=0...95	ERETA[n]	Extended Redirection Table	Section 8.2.2.8.19
0x00018000 + 0x4*n + 0x40*m, n=0...15, m=0...63	VFRSSRK[n,m]	Per Pool RSS Random Key Register	Section 8.2.2.8.20
0x00019000 + 0x4*n + 0x40*m, n=0...15, m=0...63	VFRETA[n,m]	Per Pool Redirection Table	Section 8.2.2.8.21
0x0000EC30	SYNQF	SYN Packet Queue Filter	Section 8.2.2.8.22

Table 8-4. BAR0 Registers Summary [continued]

Offset / Alias Offset	Abbreviation	Name	Section
0x00005128 + 0x4*n, n=0...7	ETQF[n]	EType Queue Filter	Section 8.2.2.8.23
0x0000EC00 + 0x4*n, n=0...7	ETQS[n]	EType Queue Select	Section 8.2.2.8.24
0x00005084	ETAG_ETYPE	E-tag EtherType Register	Section 8.2.2.8.25
0x0000507C	VXLANCTRL	VXLAN control	Section 8.2.2.8.26
Receive DMA Registers			
0x0000D000 + 0x40*n, n=0...63 and 0x00001000 + 0x40*(n-64), n=64...127	RDBAL[n]	Receive Descriptor Base Address Low	Section 8.2.2.9.1
0x0000D004 + 0x40*n, n=0...63 and 0x00001004 + 0x40*(n-64), n=64...127	RDBAH[n]	Receive Descriptor Base Address High	Section 8.2.2.9.2
0x0000D008 + 0x40*n, n=0...63 and 0x00001008 + 0x40*(n-64), n=64...127	RDLEN[n]	Receive Descriptor Length	Section 8.2.2.9.3
0x0000D010 + 0x40*n, n=0...63 and 0x00001010 + 0x40*(n-64), n=64...127	RDH[n]	Receive Descriptor Head	Section 8.2.2.9.4
0x0000D018 + 0x40*n, n=0...63 and 0x00001018 + 0x40*(n-64), n=64...127	RDT[n]	Receive Descriptor Tail	Section 8.2.2.9.5
0x00001028 + 0x40*n, n=0...63 and 0x0000D028 + 0x40*(n-64), n=64...127	RXCTL[n]	Receive Descriptor Control	Section 8.2.2.9.6
0x00001014 + 0x40*n, n=0...63 and 0x0000D014 + 0x40*(n-64), n=64...127	SRRCTL[n]	Split Receive Control Registers	Section 8.2.2.9.7
0x00002100 + 0x4*n, n=0...15	SRRCTL_ALIAS[n]	Split Receive Control Registers alias	Section 8.2.2.9.8
0x00002F00	RDRXCTL	Receive DMA Control Register	Section 8.2.2.9.9
0x00003C00 + 0x4*n, n=0...7	RXPBSIZE[n]	Receive Packet Buffer Size	Section 8.2.2.9.10
0x00003000	RXCTRL	Receive Control Register	Section 8.2.2.9.11
Transmit Registers			
0x00008100	DTXMXSZRQ	DMA Tx TCP Max Allow Size Requests	Section 8.2.2.10.1
0x00004A80	DMATXCTL	DMA Tx Control	Section 8.2.2.10.2
0x00004A88	DTXTCPFLGL	DMA Tx TCP Flags Control Low	Section 8.2.2.10.3
0x00004A8C	DTXTCPFLGH	DMA Tx TCP Flags Control High	Section 8.2.2.10.4
0x00006000 + 0x40*n, n=0...127	TDBAL[n]	Transmit Descriptor Base Address Low	Section 8.2.2.10.5
0x00006004 + 0x40*n, n=0...127	TDBAH[n]	Transmit Descriptor Base Address High	Section 8.2.2.10.6
0x00006008 + 0x40*n, n=0...127	TDLEN[n]	Transmit Descriptor Length	Section 8.2.2.10.7
0x00006010 + 0x40*n, n=0...127	TDH[n]	Transmit Descriptor Head	Section 8.2.2.10.8

Table 8-4. BAR0 Registers Summary [continued]

Offset / Alias Offset	Abbreviation	Name	Section
0x00006018 + 0x40*n, n=0...127	TDT[n]	Transmit Descriptor Tail	Section 8.2.2.10.9
0x00006028 + 0x40*n, n=0...127	TXDCTL[n]	Transmit Descriptor Control	Section 8.2.2.10.10
0x00006038 + 0x40*n, n=0...127	TDWBAL[n]	Tx Descriptor Completion Write Back Address Low	Section 8.2.2.10.11
0x0000603C + 0x40*n, n=0...127	TDWBAH[n]	Tx Descriptor Completion Write Back Address High	Section 8.2.2.10.12
0x0000CC00 + 0x4*n, n=0...7	TXPBSIZE[n]	Transmit Packet Buffer Size	Section 8.2.2.10.13
0x0000CD10	MNGTXMAP	Manageability Transmit TC Mapping	Section 8.2.2.10.14
0x00008120	MTQC	Multiple Transmit Queues Command Register	Section 8.2.2.10.15
0x00004950 + 0x4*n, n=0...7	TXPBTHRESH[n]	Tx Packet Buffer Threshold	Section 8.2.2.10.16
0x00017100	TAG_ETYPE	Tags EtherTypes	Section 8.2.2.10.17
DCB Registers			
0x00002430	RTRPCS	DCB Receive Packet plane Control and Status	Section 8.2.2.11.1
0x0000CD00	RTTPCS	DCB Transmit Packet plane Control and Status	Section 8.2.2.11.2
0x00003020	RTRUP2TC	DCB Receive User Priority to Traffic Class	Section 8.2.2.11.3
0x0000C800	RTTUP2TC	DCB Transmit User Priority to Traffic Class	Section 8.2.2.11.4
0x00002140 + 0x4*n, n=0...7	RTRPT4C[n]	DCB Receive Packet plane T4 Config	Section 8.2.2.11.5
0x000082E0 + 0x4*n, n=0...3	TXLLQ[n]	Strict Low Latency Tx Queues	Section 8.2.2.11.6
0x00004910 + 0x4*n, n=0...7	RTTDT2C[n]	DCB Transmit Descriptor Plane T2 Config	Section 8.2.2.11.7
0x0000CD20 + 0x4*n, n=0...7	RTTPT2C[n]	DCB Transmit Packet plane T2 Config	Section 8.2.2.11.8
0x00004980	RTTQCNRM	DCB Transmit QCN Rate-Scheduler MMW	Section 8.2.2.11.9
0x00004904	RTTDQSEL	DCB Transmit Descriptor Plane Queue Select	Section 8.2.2.11.10
0x00004908	RTTDT1C	DCB Transmit Descriptor Plane T1 Config	Section 8.2.2.11.11
0x00004984	RTTQCNRC	DCB Transmit QCN Rate-Scheduler Config	Section 8.2.2.11.12
0x00004988	RTTQCNR	DCB Transmit QCN Rate-Scheduler Status	Section 8.2.2.11.13
0x00004A90	RTTQCNTG	DCB Transmit QCN Tagging	Section 8.2.2.11.14
0x0000498C	RTTQCNR	DCB Transmit QCN Rate Reset	Section 8.2.2.11.15

Table 8-4. BAR0 Registers Summary [continued]

Offset / Alias Offset	Abbreviation	Name	Section
TPH Registers			
0x00002200 + 0x4*n, n=0...15	TPH_RXCTRL_ALIAS[n]	Rx TPH Control Register Alias	Section 8.2.2.12.1
0x0000D00C + 0x40*n, n=0...63 and 0x0000100C + 0x40*(n-64), n=64...127	TPH_RXCTRL[n]	Rx TPH Control Register	Section 8.2.2.12.2
0x0000600C + 0x40*n, n=0...127	TPH_TXCTRL[n]	Tx TPH Control Registers	Section 8.2.2.12.3
Timers Registers			
0x0000004C	TCPTIMER	TCP Timer	Section 8.2.2.13.1
Flow Director Registers			
0x0000EE00	FDIRCTRL	Flow Director Filters control register	Section 8.2.2.14.1
0x0000EE68	FDIRHKEY	Flow Director Filters Lookup Table HASH Key	Section 8.2.2.14.2
0x0000EE6C	FDIRSKEY	Flow Director Filters Signature HASH Key	Section 8.2.2.14.3
0x0000EE3C	FDIRDIP4M	Flow Director Filters DIPv4 Mask	Section 8.2.2.14.4
0x0000EE40	FDIRSIP4M	Flow Director Filters Source IPv4 Mask	Section 8.2.2.14.5
0x0000EE44	FDIRTCPM	Flow Director Filters TCP Mask	Section 8.2.2.14.6
0x0000EE48	FDIRUDPM	Flow Director Filters UDP Mask	Section 8.2.2.14.7
0x0000EE78	FDIRSCTPM	Flow Director Filters SCTP Mask	Section 8.2.2.14.8
0x0000EE74	FDIRIP6M	Flow Director Filters IPv6 Mask	Section 8.2.2.14.9
0x0000EE70	FDIRM	Flow Director Filters Other Mask	Section 8.2.2.14.10
0x0000EE38	FDIRFREE	Flow Director Filters Free	Section 8.2.2.14.11
0x0000EE4C	FDIRLEN	Flow Director Filters Length	Section 8.2.2.14.12
0x0000EE50	FDIRUSTAT	Flow Director Filters Usage Statistics	Section 8.2.2.14.13
0x0000EE54	FDIRFSTAT	Flow Director Filters Failed Usage Statistics	Section 8.2.2.14.14
0x0000EE58	FDIRMATCH	Flow Director Filters Match Statistics	Section 8.2.2.14.15
0x0000EE5C	FDIRMISS	Flow Director Filters Miss Match Statistics	Section 8.2.2.14.16
0x0000EE0C + 0x4*n, n=0...2	FDIRSIPV6[n]	Flow Director Filters Source IPv6	Section 8.2.2.14.17
0x0000EE18	FDIRIPSA	Flow Director Filters IP SA	Section 8.2.2.14.18
0x0000EE1C	FDIRIPDA	Flow Director Filters IP DA	Section 8.2.2.14.19
0x0000EE20	FDIRPORT	Flow Director Filters Port	Section 8.2.2.14.20
0x0000EE24	FDIRVLAN	Flow Director Filters VLAN and FLEX bytes	Section 8.2.2.14.21
0x0000EE28	FDIRHASH	Flow Director Filters Hash Signature	Section 8.2.2.14.22
0x0000EE2C	FDIRCMD	Flow Director Filters Command register	Section 8.2.2.14.23

Table 8-4. BAR0 Registers Summary [continued]

Offset / Alias Offset	Abbreviation	Name	Section
MAC Registers			
0x00004298	MAC_SGMII_BUSY	MAC SGMII Busy Register	Section 8.2.2.15.1
0x00004240	HLREG0	Highlander Control 0 Register	Section 8.2.2.15.2
0x00004244	HLREG1	Highlander Status 1 Register	Section 8.2.2.15.3
0x00004248	PAP	Pause and Pace Register	Section 8.2.2.15.4
0x0000425C	MSCA	MDI Single Command and Address	Section 8.2.2.15.5
0x00004260	MSRWD	MDI Single Read and Write Data	Section 8.2.2.15.6
0x00004268	MAXFRS	Max Frame Size	Section 8.2.2.15.7
0x000042A4	LINKS	Link Status Register	Section 8.2.2.15.8
0x000042D0	MMNGC	MAC Manageability Control Register	Section 8.2.2.15.9
0x00004330	MACC	MAC Control register	Section 8.2.2.15.10
Statistic Registers			
0x00004044	ERR_PKT_VLAN_MIS_CNT	Error Packets VLAN Mismatch Counter	Section 8.2.2.16.1
0x00004048	ERR_PKT_ADD_MIS_CNT	Error Packet Address Mismatch Counter	Section 8.2.2.16.2
0x0000403C	LINK_DN_CNT	LINK Down Counter	Section 8.2.2.16.3
0x00004018	MBSDC	Bad SFD Count	Section 8.2.2.16.4
0x00004000	CRCERRS	CRC Error Count	Section 8.2.2.16.5
0x00004004	ILLERRC	Illegal Byte Error Count	Section 8.2.2.16.6
0x00004008	ERRBPC	Error Byte Packet Count	Section 8.2.2.16.7
0x00004034	MLFC	MAC Local Fault Count	Section 8.2.2.16.8
0x00004038	MRFC	MAC Remote Fault Count	Section 8.2.2.16.9
0x00004040	RLEC	Receive Length Error Count	Section 8.2.2.16.10
0x00008780	SSVPC	Switch Security Violation Packet Count	Section 8.2.2.16.11
0x00003F60	LXONTXC	Link XON Transmitted Count	Section 8.2.2.16.12
0x000041A4	LXONRXCNT	Link XON Received Count	Section 8.2.2.16.13
0x00003F68	LXOFFTXC	Link XOFF Transmitted Count	Section 8.2.2.16.14
0x000041A8	LXOFFRXCNT	Link XOFF Received Count	Section 8.2.2.16.15
0x00003F00 + 0x4*n, n=0...7	PXONTXC[n]	Priority XON Transmitted Count	Section 8.2.2.16.16
0x00004140 + 0x4*n, n=0...7	PXONRXCNT[n]	Priority XON Received Count	Section 8.2.2.16.17
0x00003F20 + 0x4*n, n=0...7	PXOFFTXCNT[n]	Priority XOFF Transmitted Count	Section 8.2.2.16.18
0x00004160 + 0x4*n, n=0...7	PXOFFRXCNT[n]	Priority XOFF Received Count	Section 8.2.2.16.19
0x00003240 + 0x4*n, n=0...7	PXON2OFFCNT[n]	Priority XON to XOFF Count	Section 8.2.2.16.20
0x0000405C	PRC64	Packets Received [64 Bytes] Count	Section 8.2.2.16.21

Table 8-4. BAR0 Registers Summary [continued]

Offset / Alias Offset	Abbreviation	Name	Section
0x00004060	PRC127	Packets Received [65-127 Bytes] Count	Section 8.2.2.16.22
0x00004064	PRC255	Packets Received [128-255 Bytes] Count	Section 8.2.2.16.23
0x00004068	PRC511	Packets Received [256-511 Bytes] Count	Section 8.2.2.16.24
0x0000406C	PRC1023	Packets Received [512-1023 Bytes] Count	Section 8.2.2.16.25
0x00004070	PRC1522	Packets Received [1024 to Max Bytes] Count	Section 8.2.2.16.26
0x00004078	BPRC	Broadcast Packets Received Count	Section 8.2.2.16.27
0x0000407C	MPRC	Multicast Packets Received Count	Section 8.2.2.16.28
0x00004074	GPRC	Good Packets Received Count	Section 8.2.2.16.29
0x00004088	GORCL	Good Octets Received Count Low	Section 8.2.2.16.30
0x0000408C	GORCH	Good Octets Received Count High	Section 8.2.2.16.31
0x000041B0	RXNFGPC	Good Rx Non-Filtered Packet Counter	Section 8.2.2.16.32
0x000041B4	RXNFBCL	Good Rx Non-Filter Byte Counter Low	Section 8.2.2.16.33
0x000041B8	RXNFBCH	Good Rx Non-Filter Byte Counter High	Section 8.2.2.16.34
0x00002F50	RXDGPC	DMA Good Rx Packet Counter	Section 8.2.2.16.35
0x00002F54	RXDGBCL	DMA Good Rx Byte Counter Low	Section 8.2.2.16.36
0x00002F58	RXDGBCH	DMA Good Rx Byte Counter High	Section 8.2.2.16.37
0x00002F5C	RXDDPC	DMA Duplicated Good Rx Packet Counter	Section 8.2.2.16.38
0x00002F60	RXDDBCL	DMA Duplicated Good Rx Byte Counter Low	Section 8.2.2.16.39
0x00002F64	RXDDBCH	DMA Duplicated Good Rx Byte Counter High	Section 8.2.2.16.40
0x00002F68	RXLPBKPC	DMA Good Rx LPBK Packet Counter	Section 8.2.2.16.41
0x00002F6C	RXLPKBCL	DMA Good Rx LPBK Byte Counter Low	Section 8.2.2.16.42
0x00002F70	RXLPKBCH	DMA Good Rx LPBK Byte Counter High	Section 8.2.2.16.43
0x00002F74	RXDLPBKPC	DMA Duplicated Good Rx LPBK Packet Counter	Section 8.2.2.16.44
0x00002F78	RXDLPKBCL	DMA Duplicated Good Rx LPBK Byte Counter Low	Section 8.2.2.16.45
0x00002F7C	RXDLPKBCH	DMA Duplicated Good Rx LPBK Byte Counter High	Section 8.2.2.16.46
0x00004080	GPTC	Good Packets Transmitted Count	Section 8.2.2.16.47
0x00004090	GOTCL	Good Octets Transmitted Count Low	Section 8.2.2.16.48
0x00004094	GOTCH	Good Octets Transmitted Count High	Section 8.2.2.16.49
0x000087A0	TXDGPC	DMA Good Tx Packet Counter	Section 8.2.2.16.50
0x000087A4	TXDGBCL	DMA Good Tx Byte Counter Low	Section 8.2.2.16.51
0x000087A8	TXDGBCH	DMA Good Tx Byte Counter High	Section 8.2.2.16.52

Table 8-4. BAR0 Registers Summary [continued]

Offset / Alias Offset	Abbreviation	Name	Section
0x000040A4	RUC	Receive Undersize Count	Section 8.2.2.16.53
0x000040A8	RFC	Receive Fragment Count	Section 8.2.2.16.54
0x000040AC	ROC	Receive Oversize Count	Section 8.2.2.16.55
0x000040B0	RJC	Receive Jabber Count	Section 8.2.2.16.56
0x000040B4	MNGPRC	Management Packets Received Count	Section 8.2.2.16.57
0x000040B8	MNGPDC	Management Packets Dropped Count	Section 8.2.2.16.58
0x000041EC	MNGPTC	Management Packets Transmitted Count	Section 8.2.2.16.59
0x000040C0	TORL	Total Octets Received Low	Section 8.2.2.16.60
0x000040C4	TORH	Total Octets Received High	Section 8.2.2.16.61
0x000040D0	TPR	Total Packets Received	Section 8.2.2.16.62
0x000040D4	TPT	Total Packets Transmitted	Section 8.2.2.16.63
0x000040D8	PTC64	Packets Transmitted (64 Bytes) Count	Section 8.2.2.16.64
0x000040DC	PTC127	Packets Transmitted [65-127 Bytes] Count	Section 8.2.2.16.65
0x000040E0	PTC255	Packets Transmitted [128-255 Bytes] Count	Section 8.2.2.16.66
0x000040E4	PTC511	Packets Transmitted [256-511 Bytes] Count	Section 8.2.2.16.67
0x000040E8	PTC1023	Packets Transmitted [512-1023 Bytes] Count	Section 8.2.2.16.68
0x000040EC	PTC1522	Packets Transmitted [Greater than 1024 Bytes] Count	Section 8.2.2.16.69
0x000040F0	MPTC	Multicast Packets Transmitted Count	Section 8.2.2.16.70
0x000040F4	BPTC	Broadcast Packets Transmitted Count	Section 8.2.2.16.71
0x00004010	MSPDC	MAC short Packet Discard Count	Section 8.2.2.16.72
0x00004120	XEC	XSUM Error Count	Section 8.2.2.16.73
0x00002300 + 0x4*n, n=0...31	RQSMR[n]	Receive Queue Statistic Mapping Registers	Section 8.2.2.16.74
0x00002F40	RXDSTATCTRL	Rx DMA Statistic Counter Control	Section 8.2.2.16.75
0x00008600 + 0x4*n, n=0...31	TQSM[n]	Transmit Queue Statistic Mapping Registers	Section 8.2.2.16.76
0x00001030 + 0x40*n, n=0...15	QPRC[n]	Queue Packets Received Count	Section 8.2.2.16.77
0x00001430 + 0x40*n, n=0...15	QPRDC[n]	Queue Packets Received Drop Count	Section 8.2.2.16.78
0x00001034 + 0x40*n, n=0...15	QBRC_L[n]	Queue Bytes Received Count Low	Section 8.2.2.16.79
0x00001038 + 0x40*n, n=0...15	QBRC_H[n]	Queue Bytes Received Count High	Section 8.2.2.16.80
0x00008680 + 0x4*n, n=0...15	QPTC[n]	Queue Packets Transmitted Count	Section 8.2.2.16.81
0x00006030 + 0x40*n, n=0...15	QPTC_ALIAS[n]	Queue Packets Transmitted Count Alias	Section 8.2.2.16.82

Table 8-4. BAR0 Registers Summary [continued]

Offset / Alias Offset	Abbreviation	Name	Section
0x00008700 + 0x8*n, n=0...15	QBTC_L[n]	Queue Bytes Transmitted Count Low	Section 8.2.2.16.83
0x00008704 + 0x8*n, n=0...15	QBTC_H[n]	Queue Bytes Transmitted Count High	Section 8.2.2.16.84
0x00005118	FCCRC	Fiber Channel CRC Error Count	Section 8.2.2.16.85
0x00002424	FCLAST	Fiber Channel Last Error Count	Section 8.2.2.16.86
0x000041C0	B2OSPC	BMC2OS Packets Sent by BMC	Section 8.2.2.16.87
0x00002F90	B2OGPRC	BMC2OS Packets Received by Host	Section 8.2.2.16.88
0x000041C4	O2BGPTC	OS2BMC packets received by BMC	Section 8.2.2.16.89
0x000087B0	O2BSPC	OS2BMC packets transmitted by host	Section 8.2.2.16.90
0x00004180	BUPRC	Total Unicast Packets Received (BMC copy)	Section 8.2.2.16.91
0x00004184	BMPRC	BMC Total Multicast Packets Received	Section 8.2.2.16.92
0x00004188	BBPRC	Total Broadcast Packets Received (BMC copy)	Section 8.2.2.16.93
0x0000418C	BUPTC	Total Unicast Packets Transmitted (BMC copy)	Section 8.2.2.16.94
0x00004190	BMPTC	BMC Total Multicast Packets Transmitted	Section 8.2.2.16.95
0x00004194	BBPTC	Total Broadcast Packets Transmitted (BMC copy)	Section 8.2.2.16.96
0x00004198	BCRCERRS	BMC FCS Receive Errors	Section 8.2.2.16.97
0x0000419C	BXONRXC	BMC Pause XON Frames Received	Section 8.2.2.16.98
0x000041E0	BXOFFRXC	BMC Pause XOFF Frames Received	Section 8.2.2.16.99
0x000041E4	BXONTXC	BMC Pause XON Frames Transmitted	Section 8.2.2.16.100
0x000041E8	BXOFFTXC	BMC Pause XOFF Frames Transmitted	Section 8.2.2.16.101
0x000041F0	B2OSDPC	Sideband receive dropped packet count	Section 8.2.2.16.102
Wake-Up Control Registers			
0x00005800	WUC	Wake Up Control Register	Section 8.2.2.17.1
0x00005808	WUFC	Wake Up Filter Control Register	Section 8.2.2.17.2
0x00005810	WUS	Wake Up Status Register	Section 8.2.2.17.3
0x00005838	IPAV	IP Address Valid	Section 8.2.2.17.4
0x00005840 + 0x8*n, n=0...3	IP4AT[n]	IPv4 Address Table	Section 8.2.2.17.5
0x00005990 + 0x4*n, n=0...11	IP6AT_EXT[n]	IPv6 Address Table Extended	Section 8.2.2.17.6
0x00005880 + 0x4*n, n=0...3	IP6AT[n]	IPv6 Address Table	Section 8.2.2.17.7
0x00005900	WUPL	Wake Up Packet Length	Section 8.2.2.17.8
0x00005A00 + 0x4*n, n=0...31	WUPM[n]	Wake Up Packet Memory (128 Bytes)	Section 8.2.2.17.9

Table 8-4. BAR0 Registers Summary [continued]

Offset / Alias Offset	Abbreviation	Name	Section
0x00009000 + 0x10*n + 0x100*m, n=0...15, m=0...3 and 0x00009600 + 0x10*(n-16) + 0x100*m, n=16...31, m=0...3	FHFT_FILTER_DW_EVEN[n,m]	Filter DW Even	Section 8.2.2.17.10
0x00009604 + 0x10*n + 0x100*m, n=0...15, m=0...3 and 0x00009004 + 0x10*(n-16) + 0x100*m, n=16...31, m=0...3	FHFT_FILTER_DW_ODD[n,m]	Filter DW Odd	Section 8.2.2.17.11
0x00009608 + 0x10*n + 0x100*m, n=0...15, m=0...3 and 0x00009008 + 0x10*(n-16) + 0x100*m, n=16...31, m=0...3	FHFT_FILTER_MASK[n,m]	Filter Mask	Section 8.2.2.17.12
0x0000900C + 0x10*n + 0x100*m, n=0...15, m=0...3 and 0x0000960C + 0x10*(n-16) + 0x100*m, n=16...31, m=0...3	FHFT_FILTER_LENGTH[n,m]	Filter Length	Section 8.2.2.17.13
Management Filters Registers			
0x00005864	MNGONLY	Manageability Only Traffic	Section 8.2.2.18.1
0x00005010 + 0x4*n, n=0...7	MAVTV[n]	Management VLAN TAG Value	Section 8.2.2.18.2
0x00005030 + 0x4*n, n=0...7	MFUTP[n]	Management Flex UDP/TCP Ports	Section 8.2.2.18.3
0x00005190 + 0x4*n, n=0...3	METF[n]	Management Ethernet Type Filters	Section 8.2.2.18.4
0x00005820	MANC	Management Control Register	Section 8.2.2.18.5
0x00005890 + 0x4*n, n=0...7	MDEF[n]	Manageability Decision Filters	Section 8.2.2.18.6
0x00005160 + 0x4*n, n=0...7	MDEF_EXT[n]	Manageability Decision Filters Ext	Section 8.2.2.18.7
0x00005050 + 0x4*n, n=0...3	BMCIP[n]	BMC IP address Register	Section 8.2.2.18.8
0x00005060	BMCIPVAL	BMC IP Valid Register	Section 8.2.2.18.9
0x000058B0 + 0x4*n + 0x10*m, n=0...3, m=0...3	MIPAF[n,m]	Manageability IP Address Filter	Section 8.2.2.18.10
0x00005910 + 0x8*n, n=0...3	MMAL[n]	Manageability Ethernet MAC Address Low	Section 8.2.2.18.11
0x00005914 + 0x8*n, n=0...3	MMAH[n]	Manageability Ethernet MAC Address High	Section 8.2.2.18.12
0x00009400 + 0x10*n, n=0...15	FTFT_FILTER_EVEN[n]	FTFT Filter DW Even words	Section 8.2.2.18.13
0x00009404 + 0x10*n, n=0...15	FTFT_FILTER_ODD[n]	FTFT Filter DW Odd words	Section 8.2.2.18.14
0x00009408 + 0x10*n, n=0...15	FTFT_FILTER0_MASK[n]	FTFT Filter Mask	Section 8.2.2.18.15
0x0000940C + 0x10*n, n=0...15	FTFT_FILTER0_LENGTH[n]	FTFT Filter Length	Section 8.2.2.18.16

Table 8-4. BAR0 Registers Summary [continued]

Offset / Alias Offset	Abbreviation	Name	Section
Time Sync (IEEE 1588) Registers			
0x00008C0C	SYSTIMEL	System time register Low	Section 8.2.2.19.1
0x00008C10	SYSTIMEH	System time register High	Section 8.2.2.19.2
0x00008C58	SYSTIMR	System time register residue	Section 8.2.2.19.3
0x00008C14	TIMEINCA	Increment attributes register	Section 8.2.2.19.4
0x00008C18	TIMADJ	Time adjustment offset register	Section 8.2.2.19.5
0x00008C68	TSIM	Time Sync Interrupt Mask Register	Section 8.2.2.19.6
0x00008C60	TSICR	Time Sync Interrupt Cause Register	Section 8.2.2.19.7
0x00008C00	TSYNCTXCTL	Tx Time Sync Control register	Section 8.2.2.19.8
0x00008C04	TXSTMPL	Tx Timestamp Value Low	Section 8.2.2.19.9
0x00008C08	TXSTMPH	Tx Timestamp Value High	Section 8.2.2.19.10
0x00005188	TSYNCRXCTL	Rx Time Sync Control register	Section 8.2.2.19.11
0x000051E8	RXSTMPL	Rx timestamp Low	Section 8.2.2.19.12
0x000051A4	RXSTMPH	Rx timestamp High	Section 8.2.2.19.13
0x00005120	RXMTRL	Rx message type register low	Section 8.2.2.19.14
0x0000003C	TSSDP	Time Sync SDP Configuration Register	Section 8.2.2.19.15
0x00008C20	TSAUXC	TimeSync Auxiliary control register	Section 8.2.2.19.16
0x00008C24	TRGTTIMELO	Target time register 0 low	Section 8.2.2.19.17
0x00008C28	TRGTTIMEHO	Target time register 0 high	Section 8.2.2.19.18
0x00008C2C	TRGTTIMEL1	Target time register 1 low	Section 8.2.2.19.19
0x00008C30	TRGTTIMEH1	Target time register 1 high	Section 8.2.2.19.20
0x00008C34	FREQOUT0	Frequency Out 0 Control Register	Section 8.2.2.19.21
0x00008C38	FREQOUT1	Frequency Out 1 Control Register	Section 8.2.2.19.22
0x00008C3C	AUXSTMPL0	Auxiliary time stamp 0 register low	Section 8.2.2.19.23
0x00008C40	AUXSTMPH0	Auxiliary time stamp 0 register high	Section 8.2.2.19.24
0x00008C44	AUXSTMPL1	Auxiliary time stamp 1 register low	Section 8.2.2.19.25
0x00008C48	AUXSTMPH1	Auxiliary time stamp 1 register high	Section 8.2.2.19.26
Virtualization PF Registers			
0x000050B0	PFFLPL	Filter Local Packets Low	Section 8.2.2.20.1
0x000050B4	PFFLPH	Filter Local Packets High	Section 8.2.2.20.2
0x000051B0	PFVTCTL	PF Virtual Control register	Section 8.2.2.20.3
0x00004B00 + 0x4*n, n=0...63	PFMAILBOX[n]	PF Mailbox	Section 8.2.2.20.4
0x00000710 + 0x4*n, n=0...3	PFMBICR[n]	PF Mailbox Interrupt Causes Register	Section 8.2.2.20.5
0x00000720 + 0x4*n, n=0...1	PFMBIMR[n]	PF Mailbox Interrupt Mask Register	Section 8.2.2.20.6

Table 8-4. BAR0 Registers Summary [continued]

Offset / Alias Offset	Abbreviation	Name	Section
0x00000700 + 0x4*n, n=0...1	PFVFLREC[n]	PF VFLR Events Clear	Section 8.2.2.20.7
0x000051E0 + 0x4*n, n=0...1	PFVFRE[n]	PF VF Receive Enable	Section 8.2.2.20.8
0x00008110 + 0x4*n, n=0...1	PFVFTE[n]	PF VF Transmit Enable	Section 8.2.2.20.9
0x00008790	PFVMECM0	PF VM 0:31 Error Count Mask	Section 8.2.2.20.10
0x00008794	PFVMECM1	PF VM 32:63 Error Count Mask	Section 8.2.2.20.11
0x00002F04	PFQDE	PF Queue Drop Enable Register	Section 8.2.2.20.12
0x00005180 + 0x4*n, n=0...1	PFVMTXSW[n]	PF VM Tx Switch Loopback Enable	Section 8.2.2.20.13
0x00008200 + 0x4*n, n=0...7	PFVFSPOOF[n]	PFVF Anti Spoof Control	Section 8.2.2.20.14
0x00008220	PFDTXGSWC	PF DMA Tx General Switch Control	Section 8.2.2.20.15
0x00017000 + 0x4*n, n=0...63	PFVMTIR[n]	PF VM Tag Insert Register	Section 8.2.2.20.16
0x00008000 + 0x4*n, n=0...63	PFVMVIR[n]	PF VM VLAN Insert Register	Section 8.2.2.20.17
0x0000F000 + 0x4*n, n=0...63	PFVML2FLT[n]	PF VM L2Control Register	Section 8.2.2.20.18
0x0000F100 + 0x4*n, n=0...63	PFVLVF[n]	PF VM VLAN Pool Filter	Section 8.2.2.20.19
0x0000F200 + 0x4*n, n=0...127	PFVLVFB[n]	PF VM VLAN Pool Filter Bitmap	Section 8.2.2.20.20
0x0000F400 + 0x4*n, n=0...127	PFUTA[n]	PF Unicast Table Array	Section 8.2.2.20.21
0x0000F600 + 0x4*n, n=0...3	PFMRCTL[n]	PF Mirror Rule Control	Section 8.2.2.20.22
0x0000F610 + 0x4*n, n=0...7	PFMRVLAN[n]	PF Mirror Rule VLAN	Section 8.2.2.20.23
0x0000F630 + 0x4*n, n=0...7	PFMRVM[n]	PF Mirror Rule Pool	Section 8.2.2.20.24
0x00008108	LVMMC_TX	Last VM Misbehavior Cause - TX	Section 8.2.2.20.25
0x00002FA8	LVMMC_RX	Last VM Misbehavior Cause - RX	Section 8.2.2.20.26
0x00008124	LMVM_TX	Last Malicious VM - TX	Section 8.2.2.20.27
0x00002FA4	LMVM_RX	Last Malicious VM - RX	Section 8.2.2.20.28
0x00008130 + 0x4*n, n=0...3	WQBR_TX[n]	Wrong Queue Behavior Register - Tx	Section 8.2.2.20.29
0x00002FB0 + 0x4*n, n=0...3	WQBR_RX[n]	Wrong Queue Behavior Register - Rx	Section 8.2.2.20.30

Table 8-4. BAR0 Registers Summary [continued]

Offset / Alias Offset	Abbreviation	Name	Section
Power Management Registers			
0x00011708	LTRC	Latency Tolerance Reporting (LTR) Control	Section 8.2.2.21.1
0x000043A0	EEER	Energy Efficient Ethernet (EEE) Register	Section 8.2.2.21.2
0x00004380	EEE_SU	Energy Efficient Ethernet (EEE) Setup Register	Section 8.2.2.21.3
0x00004398	EEE_STAT	Energy Efficient Ethernet (EEE) STATUS	Section 8.2.2.21.4
0x000041F4	TLPIC	EEE TX LPI Count	Section 8.2.2.21.5
0x000041F8	RLPIC	EEE RX LPI Count	Section 8.2.2.21.6
Manageability (ARC Subsystem) HOST Interface Registers			
0x00015F70	SWSM	Software Semaphore Register	Section 8.2.2.22.1
0x00015F74	FWSM	Firmware Semaphore Register	Section 8.2.2.22.2
0x00015F78	SW_FW_SYNC	Software-Firmware Synchronization	Section 8.2.2.22.3
0x00015F00	HICR_HOST	HOST Interface Control Register	Section 8.2.2.22.4
0x00015800 + 0x4*n, n=0...447	ARCRAM[n]	Host ARC Data RAM	Section 8.2.2.22.5
0x00015F0C	FWSTS	legacy Firmware Status register	Section 8.2.2.22.6
Manageability Extension Registers			
0x00083300 + 0x4*n, n=0...11	GL_MNG_ECDSA_PUBKEY[n]	Device Public Key	Section 8.2.2.23.1
0x00082000 + 0x4*n, n=0...15	GL_HIDA[n]	Tools Mailbox HOST Interface Descriptor Area	Section 8.2.2.23.2
0x00085000 + 0x4*n, n=0...15	PF_HIDA[n]	Per-PF AQ Mailbox HOST Interface Descriptor Area	Section 8.2.2.23.3
0x00083148	GL_MNG_SHA_EXTEND_STATUS	Measurement Status	Section 8.2.2.23.4
0x00081000 + 0x4*n, n=0...1023	GL_HIBA[n]	Tools Mailbox HOST Interface Buffer Area	Section 8.2.2.23.5
0x00083340 + 0x4*n, n=0...11	GL_MNG_SHA_EXTEND[n]	NVM Bank Measurement	Section 8.2.2.23.6
0x00083100	GL_FWRESETCNT	Firmware Reset Count	Section 8.2.2.23.7
0x000830EC	GL_MNG_FW_RAM_STAT	MNG FW RAM status registers	Section 8.2.2.23.8
0x00084000 + 0x4*n, n=0...1023	PF_HIBA[n]	Per-PF AQ Mailbox HOST Interface Buffer Area	Section 8.2.2.23.9
0x00082040	GL_HICR	Tools Mailbox HOST Interface Control Register	Section 8.2.2.23.10
0x000832C0 + 0x4*n, n=0...11	GL_MNG_SHA_EXTEND_ROM[n]	Mini-Loader Measurement	Section 8.2.2.23.11
0x00082044	GL_HICR_EN	Tools Mailbox HOST Interface Enable Register	Section 8.2.2.23.12
0x00082048	PF_HICR	AQ Mailbox HOST Interface Control Register	Section 8.2.2.23.13
0x00083044	GLDCB_GENC	Global DCB General Control	Section 8.2.2.23.14

Table 8-4. BAR0 Registers Summary [continued]

Offset / Alias Offset	Abbreviation	Name	Section
0x00083000	PRTDCB_GENC	Port DCB General Control	Section 8.2.2.23.15
0x00083020	PRTDCB_GENS	Port DCB General Status	Section 8.2.2.23.16
0x00083120	PRTGEN_CNF	General Port Configuration	Section 8.2.2.23.17
Fleep Registers			
0x000B6008	GLNVM_ULD	Unit Load Status	Section 8.2.2.24.1
0x000B6154	GLNVM_ULT	Unit Load Timeout	Section 8.2.2.24.2
0x000B6140	GLNVM_ALTIMERS	Autoload Timers	Section 8.2.2.24.3
0x000B6100	GLNVM_GENS	Global NVM General Status Register	Section 8.2.2.24.4
0x000B6108	GLNVM_FLA	Flash Access Register	Section 8.2.2.24.5
0x000B612C	GLGEN_STAT	Global Status	Section 8.2.2.24.6
0x000B6010 + 0x4*n, n=0...59	GLNVM_PROTCSR[n]	Protected CSR List	Section 8.2.2.24.7
0x000B6180 + 0x4*n, n=0...9	GENERAL_MNG_FW_DBG_CSR[n]	General FW Debug Registers	Section 8.2.2.24.8
0x000B6130	GL_MNG_HWARB_CTRL	Hardware Arbitration Control	Section 8.2.2.24.9
0x000B6134	GL_MNG_FWSM	Firmware Status	Section 8.2.2.24.10
IOSB SB Registers			
0x000A400C	GL_UFUSE_SOC	SKU Fuses	Section 8.2.2.25.1
Diagnostic Registers			
0x0008204C	GLOBAL_FW_CTL	global control and status	Section 8.2.2.26.1
VF Registers Mapping in the PF Space			
0x00000300 + 0x4*n, n=0...63	VFCTRL[n]	VF Control Register	Section 8.2.2.27.1
0x00004C00 + 0x4*n, n=0...63	VFMAILBOX[n]	VF Mailbox	Section 8.2.2.27.2
0x00013000 + 0x4*n + 0x40*m, n=0...15, m=0...63	PFMBMEM[n,m]	PF Mailbox Memory	Section 8.2.2.27.3
0x00000B00 + 0x4*n, n=0...63	VFEICR[n]	VF Extended Interrupt Cause	Section 8.2.2.27.4
0x00000C00 + 0x4*n, n=0...63	VFEICS[n]	VF Extended Interrupt Cause Set	Section 8.2.2.27.5
0x00000D00 + 0x4*n, n=0...63	VFEIMS[n]	VF Extended Interrupt Mask Set/ Read	Section 8.2.2.27.6
0x00000E00 + 0x4*n, n=0...63	VFEIMC[n]	VF Extended Interrupt Mask Clear	Section 8.2.2.27.7
0x00004D00 + 0x4*n, n=0...63	VFEIAM[n]	VF Extended Interrupt Auto Mask Enable	Section 8.2.2.27.8
0x00012500 + 0x4*n, n=0...63	VFIVAR[n]	VF Interrupt Vector Allocation Registers	Section 8.2.2.27.9
0x00004E00 + 0x4*n, n=0...63	VFIVAR_MISC[n]	VF Interrupt Vector Allocation Registers Misc	Section 8.2.2.27.10

Table 8-4. BAR0 Registers Summary [continued]

Offset / Alias Offset	Abbreviation	Name	Section
0x00003400 + 0x4*n, n=0...63	VFMRQC[n]	VF Multiple Receive Queues Command Register	Section 8.2.2.27.11
0x00008300 + 0x4*n, n=0...63	VFGPTC[n]	VF Good Packets Transmitted Count	Section 8.2.2.27.12
0x00008400 + 0x8*n, n=0...63	VFGOTC_LSB[n]	VF Good Octets Transmitted Count LSB	Section 8.2.2.27.13
0x00008404 + 0x8*n, n=0...63	VFGOTC_MSB[n]	VF Good Octets Transmitted Count MSB	Section 8.2.2.27.14
0x00001020 + 0x40*n, n=0...63	VFGORC_LSB[n]	VF Good Octets Received Count Low	Section 8.2.2.27.15
0x0000101C + 0x40*n, n=0...63	VFGPRC[n]	VF Good Packets Received Count	Section 8.2.2.27.16
0x0000D020 + 0x40*n, n=0...63	VFGORC_MSB[n]	VF Good Octets Received Count High	Section 8.2.2.27.17
0x0000D01C + 0x40*n, n=0...63	VFMPRC[n]	VF Multicast Packets Received Count	Section 8.2.2.27.18

8.2.2 Detailed Register Description - PF BAR0

8.2.2.1 General Control Registers

8.2.2.1.1 Device Control Register - CTRL (0x00000000)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	1:0	0x0	RSV	NA	Reserved.
PCIE_LEAD_DISABLE	2	0b	RW	NA	When set, the device blocks new lead requests, including manageability requests, by using this function. Once no lead requests are pending by using this function, the PCIE_Lead_Enable_Status bit is cleared. Note: After doing any change to this bit the host must read that the bit has been modified as expected before reading STATUS.PCIE_LEAD_ENABLE_STATUS bit.
LRST	3	0b	RW	NA	Link Reset. This bit performs a reset of the Controller (same as "RST") including the Ethernet Link. (MAC & PHY) resulting in a state nearly approximating the state following a power-up reset or internal P-IOSF reset, except for the system PCI configuration. Normally 0b, writing 1b initiates the reset. This bit is self-clearing. Also referred to as MAC reset. See note below for required software flow.
RESERVED	25:4	0x0	RSV	NA	Reserved.
RST	26	0b	RW	NA	Device Reset. This bit performs a complete reset of the controller (not include the MAC-PHY Ethernet link), resulting in a state nearly approximating the state following a power-up reset or internal P-IOSF reset, except for the system PCI configuration. Normally 0b, writing 1b initiates the reset. This bit is self-clearing. Also referred to as a software reset. See note below for required software flow.
RESERVED	31:27	0x0	RSV	NA	Reserved.

Note: To ensure that a global device reset has fully completed and that the device responds to subsequent accesses, programmers must wait approximately 1 ms after setting before attempting to check if the bit has cleared or to access (read or write) any other device register.

8.2.2.1.2 Device Status Register - STATUS (0x00000008)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	1:0	0x0	RSV	NA	Reserved.
LAN_ID	3:2	0x0	RO	NA	LAN ID. Provides software a mechanism to determine the device LAN identifier for this MAC. Read as: [0,0] LAN 0; [0,1] LAN 1.
PFVFSTAT	6:4	0x0	RW	NA	PFVF Status Indication. This read/write field can be used by the PF driver to pass any status indication to the VFs.
LINKUP	7	0b	RW	NA	Linkup Status Indication. This bit is useful for IOV mode. The PF software driver sets it according to the LINKS register and PHY state. It is reflected in the VFSTATUS register indicating link up to the VF drivers.
RESERVED	9:8	0x0	RSV	NA	Reserved.
NUM_VFS	17:10	0x0	RO	NA	Num VFs. This field reflects the value of the Num VFs in the IOV capability structure. Note: Bit 17 is always 0b.
IOV_ACTIVE	18	0b	RO	NA	IOV Active. This bit reflects the value of the VF Enable (VFE) bit in the IOV Control/Status register.
PCIE_LEAD_ENABLE_STATUS	19	1b	RO	NA	This is a status bit of the appropriate CTRL.PCIE_Lead_Disable bit. 1b = Associated LAN function can issue Lead requests. 0b = Associated LAN function does not issue any Lead request and all previously issued requests are complete.
RESERVED	31:20	0x0	RSV	NA	Reserved. Reads as 0b.

8.2.2.1.3 Extended Device Control Register - CTRL_EXT (0x00000018)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	13:0	0x00	RSV	NA	Reserved.
PFRSTD	14	0b	SC	NA	PF Reset Done. When set, the RSTI bit in all the VFMailbox registers are cleared and the RSTD bit in all the VFMailbox registers are set.
RESERVED	16:15	0x0	RSV	NA	Reserved.
RO_DIS	17	0b	RW	NA	Relaxed Ordering Disable. When set to 1b, the device does not request any relaxed ordering transactions. When this bit is cleared and the Enable Relaxed Ordering bit in the Device Control register is set, the device requests relaxed ordering transactions per queues as configured in the TPH_RXCTRL[n] and TPH_TXCTRL[n] registers.
RESERVED	25:18	0x0	RSV	NA	Reserved.
EXTENDED_VLAN	26	0b	RW	NA	Extended VLAN. When set, all incoming Rx packets are expected to have at least one VLAN with the Ether type as defined in the EXVET register. This bit should only be reset by a PCIe reset and should only be changed while Tx and Rx processes are stopped. Should be set to the same value as DMATXCTL.GDV

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
LINK_ENA	27	1b	RW	NA	The link on the external PHY is enabled by the embedded firmware only when the Link_Ena is set. Changing the value of the Link_Ena generates an internal interrupt to the firmware.
DRV_LOAD	28	0b	RW	NA	Software Device Driver loaded and the corresponding network interface is enabled. This bit should be set by the software device driver after it was loaded and cleared when it unloads or at PCIe reset. Changing the value of this bit generates an interrupt to the Manageability controller (MC). As a response, the MC reads this bit as an indication that the software device driver successfully loaded.
RESERVED	31:29	0x0	RSV	NA	Reserved.

8.2.2.1.4 Extended SDP Control - ESDP (0x00000020)

Reset source: LAN Power Good Reset, NVM Load: Dedicated, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
SDP0_DATA	0	0b	RW	NA	SDP0 Data Value. Used to read (write) a value of the software-controlled I/O pin SDP0. If SDP0 is configured as an output (SDP0_IODIR = 1b), this bit controls the value driven on the pin. If SDP0 is configured as an input, all reads return the current value of the pin. See Notes 1 and 2.
SDP1_DATA	1	0b	RW	NA	SDP1 Data Value. Used to read (write) a value of the software-controlled I/O pin SDP1. If SDP1 is configured as an output (SDP1_IODIR = 1b), this bit controls the value driven on the pin. If SDP1 is configured as an input, all reads return the current value of the pin. See Note 1
RESERVED	7:2	0x0	RSV	NA	Reserved.
SDP0_IODIR	8	0b	RW	NA	SDP0 Pin Directionality. Controls whether or not software-controlled pin SDP0 is configured as an input or output. 0b = Input. 1b = Output. See Notes 1 and 2.
SDP1_IODIR	9	0b	RW	NA	SDP1 Pin Directionality. Controls whether or not software-controlled pin SDP1 is configured as an input or output. 0b = Input. 1b = Output. See Note 1
RESERVED	15:10	0x0	RSV	NA	Reserved.
SDP0_NATIVE	16	0b	RW	NA	SDP0 Operating Mode. 0b = Generic software controlled I/O by SPD0_DATA and SDP0_IODIR. 1b = Native mode operation (connected to hardware function) is IEEE 1588 functionality. See Note 2.

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
SDP1_NATIVE	17	0b	RW	NA	SDP1 Operating Mode. 0b = Generic software controlled I/O by SPD1_DATA and SPD1_IODIR. 1b = Native mode operation (connected to hardware function) according to the SPD1_Function bit. See Note 1
RESERVED	24:18	0x0	RSV	NA	Reserved.
SDP1_FUNCTION	25	0b	RW	NA	SDP1 Native Mode Functionality (SDP1_NATIVE = 1b). 0b = 1588 functionality. SPD1_IODIR should be configured as an output. 1b = Reserved
RESERVED	31:26	0x0	RSV	NA	Reserved.

Notes:

1. Initial values are read from the NVM. These Fields can not be modified by the host software.
2. It is assumed that bit 15 of NC-SI Configuration 2 word in the NVM is cleared; otherwise, the SDP0 pin of function 0 is used as input pins that encode the NC-SI Package ID of the controller.
3. If bit SDP_FUNC_OFF_EN in the NVM Control word 2 is set, then SDP1 pin is strapped during PE_RST_N to determine if the PCIe function is enabled or disabled.

8.2.2.1.5 MAC GPIO Register - MAC_GPIO (0x00000030)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MAC_GPIO	3:0	0x0	RW	NA	MAC-to-PHY GPIO. Used to set the four internal MAC-to-PHY general purpose signals.
RESERVED	31:4	0x0	RSV	NA	Reserved.

8.2.2.1.6 LED Control - LEDCTL (0x00000200)

Reset source: Master Reset, NVM Load: Dedicated, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
LED0_MODE	3:0	0x0	RW	NA	LED0 Mode. This field specifies the control source for the LED0 output. An initial value of 0000b selects the LINK_UP indication.
RESERVED	4	0b	RSV	NA	Reserved.
GLOBAL_BLINK_MODE	5	0b	RW	NA	Global Blink Mode. This field specifies the blink mode of all LEDs. 0b = Blink at 200 ms on and 200 ms off. 1b = Blink at 83 ms on and 83 ms off.
LED0_IVRT	6	0b	RW	NA	LED0 Invert. This field specifies the polarity/inversion of the LED0 source prior to output or blink control. By default the output drives the cathode of the LED0 so when the LED output is 0b the LED0 is on. 0b = LED0 output is active low. 1b = LED0 output is active high.

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
LED0_BLINK	7	0b	RW	NA	LED0 Blink. This field specifies whether or not to apply blink logic to the (inverted) LED0 control source prior to the LED0 output. 0b = Do not blink LED0 output. 1b = Blink LED0 output.
LED1_MODE	11:8	0x1	RW	NA	LED1 Mode. This field specifies the control source for the LED1 output. An initial value of 0001b selects the 10 Gb/s link indication.
RESERVED	13:12	0x0	RSV	NA	Reserved.
LED1_IVRT	14	0b	RW	NA	LED1 Invert. This field specifies the polarity/inversion of the LED1 source prior to output or blink control. By default the output drives the cathode of the LED so when the LED1 output is 0b the LED1 is on. 0b = LED1 output is active low. 1b = LED output is active high.
LED1_BLINK	15	1b	RW	NA	LED1 Blink. This field specifies whether or not to apply blink logic to the (inverted) LED1 control source prior to the LED output. 0b = Do not blink LED output. 1b = Blink LED output.
LED2_MODE	19:16	0x4	RW	NA	LED2 Mode. This field specifies the control source for the LED2 output. An initial value of 0100b selects LINK/ACTIVITY indication.
RESERVED	21:20	0x0	RSV	NA	Reserved.
LED2_IVRT	22	0b	RW	NA	LED2 Invert. This field specifies the polarity/inversion of the LED2 source prior to output or blink control. By default the output drives the cathode of the LED2 so when the LED2 output is 0b the LED2 is on. 0b = LED2 output is active low. 1b = LED2 output is active high.
LED2_BLINK	23	0b	RW	NA	LED2 Blink. This field specifies whether or not to apply blink logic to the (inverted) LED2 control source prior to the LED2 output. 0b = Do not blink LED2 output. 1b = Blink LED2 output.
LED3_MODE	27:24	0x5	RW	NA	LED3 Mode. This field specifies the control source for the LED3 output. An initial value of 0101b selects the 1 Gb/s link indication.
RESERVED	29:28	0x0	RSV	NA	Reserved.
LED3_IVRT	30	0b	RW	NA	LED3 Invert. This field specifies the polarity/inversion of the LED3 source prior to output or blink control. By default the output drives the cathode of the LED3 so when the LED3 output is 0b the LED3 is on. 0b = LED3 output is active low. 1b = LED3 output is active high.
LED3_BLINK	31	0b	RW	NA	LED3 Blink. This field specifies whether or not to apply blink logic to the (inverted) LED3 control source prior to the LED3 output. 0b = Do not blink LED3 output. 1b = Blink LED3 output.

Note: All init bits in this register are read from the NVM.

8.2.2.1.7 LED Status - LEDSTAT (0x00000204)

Reset source: N/A, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
LEDSTAT	3:0	0x0	RO	NA	Reflects LED[3:0] state
RESERVED	31:4	0x0	RSV	NA	

8.2.2.1.8 Extended VLAN Ether Type - Receive - EXVET (0x00005078)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	15:0	0x00	RSV	NA	Reserved.
VET_EXT	31:16	0x8100	RW	NA	Outer VLAN Ether Type. The VLAN Tag Protocol Identifier (TPID). Note: This field appears in little endian (MS byte first on the wire).

8.2.2.1.9 Extended VLAN Ether Type - Transmit - EXVET_T (0x00008224)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	15:0	0x00	RSV	NA	Reserved.
VET_EXT	31:16	0x8100	RW	NA	Outer VLAN Ether Type. The VLAN Tag Protocol Identifier (TPID). Note: This field appears in little endian (MS byte first on the wire).

8.2.2.1.10 General Receive Control - GRC (0x00015F64)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	0	0b	RSV	NA	
APME	1	0b	RW	NA	APM Wake Enable of Function 0 is loaded from the "APM Enable Port 0" bit in NVM Control Word 3 APM Wake Enable of Function 1 is loaded from the "APM Enable Port 1" bit in NVM Control Word 3
RESERVED	31:2	0x0	RSV	NA	Reserved.

8.2.2.1.11 SFP I2C Command - I2CCMD (0x00015F58)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
DATA	15:0	X	RW	NA	Data in a write command, software places the data bits and then the MAC shifts them out to the I2C bus. Data in a read command, the MAC reads these bits serially from the I2C bus and then software reads them from this location. This field is interpreted as two consecutive bytes. When I2CPARAMS.ACCESS_WIDTH = 0, bits 15:8 are not used.
REGADD	23:16	0x0	RW	NA	I2C Register Address. For example, register 0, 1, 2... 255.
RESERVED	26:24	0x0	RSV	NA	Reserved
OP	27	0b	RW	NA	Op Code. 0b = I2C write. 1b = I2C read.
RESET	28	0b	RW	NA	Reset Sequence. If set, sends a reset sequence before the actual read or write. This bit is self clearing. A reset sequence is defined as nine consecutive stop conditions.
R	29	1b	RW	NA	Ready Bit. Set to 1b by the device at the end of the I2C transaction. Indicates the read or write completed. Reset by a software write of a command.
RESERVED	30	0b	RSV	NA	Reserved
E	31	0b	RW	NA	Error. This bit set is to 1b by hardware when it fails to complete an I2C read. Reset by a software write of a command. Note: This bit is valid only when the Ready (R) bit is set.

8.2.2.1.12 SFP I2C Parameters - I2CPARAMS (0x00015F5C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
WRITE_TIME	4:0	0x6	RW	NA	Write Time. Defines the delay between a write access and the next access. The value is in milliseconds. Note: A value of zero is not valid.
READ_TIME	7:5	0x2	RW	NA	Read Time. Defines the delay between a read access and the next access. The value is in microseconds. Note: A value of zero is not valid.
I2CBB_EN	8	0b	RW	NA	I2C Bit-bang Enable. If set, the I2C_CLK and I2C_DATA lines are controlled via the CLK, DATA, and DATA_OE_N fields of this register. Otherwise, they are controlled by hardware once activated via the I2CCMD register.
CLK_OUT	9	0b	RW	NA	I2C Clock. While in bit-bang mode, controls the value driven on the I2C_CLK pad of this port.
DATA_OUT	10	0b	RW	NA	I2C_DATA. While in bit-bang mode and when the DATA_OE_N field is zero, controls the value driven on the I2C_DATA pad of this port.

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
DATA_OE_N	11	0b	RW	NA	I2C_DATA_OE_N. While in bit-bang mode, controls the direction of the I2C_DATA pad of this port. 0b = Pad is output. 1b = Pad is input.
DATA_IN	12	0b	RO	NA	I2C Data In. Reflects the value of the I2C_DATA pad. While in bit-bang mode when the DATA_OE_N field is zero, this field reflects the value set in the DATA_OUT field.
CLK_OE_N	13	0b	RW	NA	I2C Clock Output Enable. While in bit-bang mode, controls the direction of the I2C_CLK pad of this port. 0b = Pad is output. 1b = Pad is input.
CLK_IN	14	0b	RO	NA	I2C Clock In Value. Reflects the value of the I2C_CLK pad. While in bit-bang mode when the CLK_OE_N field is zero, this field reflects the value set in the CLK_OUT field.
CLK_STRETCH_DIS	15	0b	RW	NA	Clock Stretch Disable. 0b = Enable slave clock stretching support in an I2C access. 1b = Disable clock stretching support in an I2C access.
ACCESS_WIDTH	16	0b	RW	NA	I2C Access width: 0b: Byte access 1b: Word access
RESERVED	23:17	0x0	RSV	NA	Reserved. Write 0x0, ignore on read.
PHYADD	30:24	0x0	RW	NA	Device Address Bits 7:1. The actual address used is b{PHYADD[6:0], 0}.
I2C_DATA_ORDER	31	0b	RW	NA	I2C Data Order. 0b = I2CCMD.DATA field read in byte order. 1b = I2CCMD.DATA field read in word order.

Note: This register is reset only on LAN_PWR_GOOD.

8.2.2.1.13 Device and Functions Enable Control - DEV_FUNC_EN (0x00015FF0)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
LAN_PCI_DISABLE	0	0b	RO	NA	LAN PCI Disable. When set, one LAN port is disabled. The function that is disabled is determined by the LAN Disable Select bit. If the disabled function is function 0, it acts as a dummy function or the other LAN function depending on the Dummy function enable setting. If the disabled port is used for WoL or by MC, only the DMA block of the port is powered down. Otherwise, the port is powered down including the PHY.
LAN_DISABLE_SELECT	1	0b	RO	NA	Value of NVM control Word2 bit 9
DEV_OFF_EN	2	0b	RO	NA	Value of NVM control Word2 bit 10
RESERVED	3				Reserved

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
IGNORE_PHY_FW_VALID	4	0b	RO	NA	Value of NVM control Word2 bit 12
RESERVED	31:5	0x0	RSV	NA	Reserved

8.2.2.1.14 Function Active and Power State to Manageability - FACTPS (0x00015FEC)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FUNC0_POWER_STATE	1:0	0x0	RO	NA	Function 0 Power State Power state indication of function 0. 00b = DR 01b = D3 10b = D0a 11b = D0u
LAN0_VALID	2	0b	RO	NA	Shadow of Enable Pin
FUNC0_AUX_EN	3	0b	RO	NA	Enable bit shadow from the configuration space.
DEV_DIS_N	4	0b	RO	NA	Reflects the status of IOR dev dis
RESERVED	5	0b	RSV	NA	Reserved.
FUNC1_POWER_STATE	7:6	0x0	RO	NA	Function 1 Power State Power state indication of function 0. 00b = DR 01b = D3 10b = D0a 11b = D0u
LAN1_VALID	8	0b	RO	NA	Shadow of Enable Pin
FUNC1_AUX_EN	9	0b	RO	NA	Enable bit shadow from the configuration space
RESERVED	29:10	0x0	RSV	NA	Reserved.
LAN_FUNCTION_SEL	30	0b	RO	NA	Value of Legacy NVM Control Word2 bit 15
RESERVED	31	0b	RSV	NA	

Note: In regular mode (port swap disabled - LAN_FUNCTION_SEL = 0), the power state indication of port 0 is mapped to the FUNC0_POWER_STATE field and the power state indication of port 1 is mapped the FUNC1_POWER_STATE field. And vice versa when port swap mode is enabled (LAN_FUNCTION_SEL = 1).

8.2.2.2 NVM Registers

8.2.2.2.1 EEPROM Mode Control Register - EEC (0x00015FF8)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RSV	7:0	0x0	RSV	NA	Reserved.
EE_PRES	8	0b	RO	NA	NVM Present (set by FW) When this bit is set, indicates that an NVM is present and has the correct signature field.
AUTO_RD	9	0b	RO	NA	NVM Auto-Read Done. When set to 1b, this bit indicates that the auto-read by from the NVM caused by the latest reset for this port is done. This bit is also set when the NVM is not present or when its signature field is not valid.
MNG_READY	10	0b	RO	NA	Management autoload Done. When set, indicates the sections needed for manageability are valid.
EE_SIZE	14:11	0x7	RO	NA	NVM Size Via EEPROM Mode. This field defines the size of the NVM that is accessible via EEPROM mode. This is equal to the size of the internal shadow RAM, fixed to 16 KB (0x7).
RSV	16:15	0x0	RSV	NA	Reserved.
PCI__GENERAL_DONE	17	0b	RO	NA	PCIe General Done. When set to 1b, indicates that the PCIe general section read from the NVM is done. This bit is cleared when auto-read starts. This bit is also set when the NVM is not present or when its signature field is not valid.
PCI_FUNC_DONE	18	0b	RO	NA	PCIe Function Done. When set to 1b, indicates that the PCIe function section read from the NVM is done. This bit is cleared when auto-read starts. This bit is also set when the NVM is not present or when its signature field is not valid. Note: This bit returns the relevant done indication for the function that reads the register.
CORE_DONE	19	0b	RO	NA	When set to 1b, indicates that the core section read from the NVM is done. This bit is cleared when auto-read starts. This bit is also set when the NVM is not present or when its signature field is not valid. Note: This bit returns the relevant done indication for the function that reads the register.
CORE_CSR_DONE	20	0b	RO	NA	When set to 1b, indicates that the core CSR section read from the NVM is done. This bit is cleared when auto-read starts. This bit is also set when the NVM is not present or when its signature field is not valid. Note: This bit returns the relevant done indication for the function that reads the register.

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MAC_DONE	21	0b	RO	NA	When set to 1b, indicates that the MAC section read from the NVM is done. This bit is cleared when auto-read starts. This bit is also set when the NVM is not present or when its signature field is not valid. Note: This bit returns the relevant done indication for the function that reads the register.
HIP_DONE	22	0b	RO	NA	When set, indicates that the HIP section read from the NVM is done. This bit is cleared when auto-read starts. This bit is also set when the NVM is not present or when its signature field is not valid.
RESERVED	23				Reserved
RSV	24	0b	RSV	NA	Reserved.
SEC1VAL	25	0b	RO	NA	Sector 1 Valid. When set to 1b, indicates that the content of the second shadow RAM section (from byte address 0x4000 to 0x7FFF) of the Flash device is valid. When set to 0b, indicates that the content of the first section (from byte address 0x0000 to 0x3FFF) is valid. Meaningful only when the EE_PRES bit is read as 1b.
FLUDONE	26	0b	WO	NA	Flash Update Done. When set to 1b, indicates that the last Flash update process completed. This bit is set by firmware, and auto clears when EEC.FLUPD is set.
RSV	31:27	0x0	RSV	NA	Reserved.

8.2.2.2.2 Legacy Flash Access Register - FLA (0x00015F68)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	5:0	0x0	RSV	NA	
LOCKED	6	0b	RO	NA	When set to 1b, this bit indicates that the NVM is in secure mode. When set to 0b, the NVM is in non-secure mode.
NVM_SEC_DIS	7	0b	RO	NA	Reflects the value of the NVM_SEC_DIS strap
RESERVED	31:8	0x0	RSV	NA	Reserved. Reads as 0b.

Note: When NVM is locked, this register is RO to Software.

8.2.2.3 Flow Control Registers

8.2.2.3.1 MAC Flow Control Register - MFLCN (0x00004294)

Reset source: MAC Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PMCF	0	0b	RW	NA	Pass MAC Control Frames. Filter Out Unrecognized Pause (Flow Control Opcode doesn't match) And Other Control Frames 0 = Filter Unrecognized Pause Frames 1 = Pass/forward Unrecognized Pause Frames
DPF	1	0b	RW	NA	Discard Pause Frame When set to 0b, Pause frames are sent to the host. Setting this bit to 1b causes PAUSE frames to be discarded only when RFCE or RPFCE are set to 1b. If both RFCE and RPFCE are set to 0b, this bit has no effect on incoming PAUSE frames.
RPFCE	2	0b	RW	NA	Receive Priority Flow Control Mode Indicates that the MAC responds to the reception of Priority Flow Control packets. If auto negotiation is enabled this bit should be set by software to the negotiated flow control value. This bit shall be set as a logical "OR" over the RPFCE[7:0] bitmap. It is useful to control forwarding of PFC frames to host if required. Notes: - Priority Flow control should be enabled in DCB mode only. - Receive Priority Flow Control and Receive Link Flow Control are mutually exclusive and user should not configure both of them to be enabled at the same time. - This bit should not be set if bit 3 is set.
RFCE	3	0b	RW	NA	Receive Link Flow Control Enable. If auto negotiation is enabled, this bit should be set by software to the negotiated flow control value. Note: This bit should not be set if bit 2 is set.
RPFCE	11:4	0x0	RW	NA	Receive Priority Flow Control Enable bitmap When bit n is set, the device responds to the reception of Priority Flow Control packets for UP#n. When bit n is cleared, Priority Flow Control indication received for UP#n is ignored and transmit from UP#n will not be paused.
RESERVED	31:12	0x0	RSV	NA	Reserved.

8.2.2.3.2 Priority Flow Control Type Opcode - PFCTOP (0x0000431C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FCT	15:0	0x8808	RW	NA	Priority Flow Control EtherType. Note: This field appears in little endian (MS byte first on the wire).
FCOP	31:16	0x0101	RW	NA	Priority Flow Control Opcode. Note: This field appears in big endian (LS byte first on the wire).

Note: This register contains the Type and Opcode fields that are matched against a recognized priority flow control packet.

8.2.2.3.3 Flow Control Transmit Timer Value n - FCTTVN[n] (0x00003200 + 0x4*n, n=0...3)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TTV_2N	15:0	0x0	RW	NA	Transmit Timer Value 2n. Timer value included in XOFF frames as Timer (2n). The same value must be set to User Priorities (UPs) attached to the same TC, as defined in the RTTUP2TC register. For legacy 802.3X flow control packets, TTV0 is the only timer that is used.
TTV_2N_1	31:16	0x0	RW	NA	Transmit Timer Value 2n+1. Timer value included in XOFF frames as Timer 2n+1. The same value must be set to UPs attached to the same TC, as defined in the RTTUP2TC register.

Notes:

- The 16-bit value in the TTV field is inserted into a transmitted frame (either XOFF frames or any pause frame value in any software transmitted packets). It counts in units of slot time (usually 64 bytes).
- The device uses a fixed slot time value of 64-byte times.

8.2.2.3.4 Flow Control Receive Threshold Low - FCRTL[n] (0x00003220 + 0x4*n, n=0...7)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	4:0	0x0	RSV	NA	Reserved.
RTL	18:5	0x0	RW	NA	Receive Threshold Low n. Receive packet buffer n FIFO low water mark for flow control transmission (32 bytes granularity).
RESERVED	30:19	0x0	RSV	NA	Reserved.
XONE	31	0b	RW	NA	XON Enable n. Per the receive packet buffer XON enable. 0b = Disabled. 1b = Enabled.

Note: This register contains the receive threshold used to determine when to send an XON packet and counts in units of bytes. The lower four bits must be programmed to 0x0 (16-byte granularity). Software must set XONE to enable the transmission of XON frames. Each time incoming packets cross the receive high threshold (become more full), and then crosses the receive low threshold, with XONE enabled (1b), hardware transmits an XON frame.

8.2.2.3.5 Flow Control Receive Threshold High - FCRTH[n] (0x00003260 + 0x4*n, n=0...7)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	4:0	0x0	RSV	NA	Reserved.
RTH	18:5	0x0	RW	NA	Receive Threshold High n. Receive packet buffer n FIFO high water mark for flow control transmission (32 bytes granularity).
RESERVED	29:19	0x0	RSV	NA	Reserved.
RESERVED	30				Reserved
FCEN	31	0b	RW	NA	Transmit Flow control enable for packet buffer n. Should be set only for traffic classes to which UPs are mapped (in RTRUP2TC register).

Note: This register contains the receive threshold used to determine when to send an XOFF packet and counts in units of bytes. This value must be at least eight bytes less than the maximum number of bytes allocated to the receive packet buffer and the lower five bits must be programmed to 0x0 (32-byte granularity). Each time the receive FIFO reaches the fullness indicated by RTH, hardware transmits a pause frame if the transmission of flow control frames is enabled.

8.2.2.3.6 Flow Control Refresh Threshold Value - FCRTV (0x000032A0)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FC_REFRESH_TH	15:0	0x0	RW	NA	Flow Control Refresh Threshold. This value is used to calculate the actual refresh period for sending the next pause frame if conditions for a pause state are still valid (buffer fullness above low threshold value). The formula for the refresh period for priority group N is: $FCTTV[N/2].TTV[Nmod2] - FCRTV.FC\_refresh\_th$ Note: The FC_refresh_th must be smaller than TTV of the TC and larger than the maximum packet size in the TC + FC packet size + link latency and Tx latency and Rx latency in 64-byte units.
RESERVED	31:16	0x0	RSV	NA	Reserved.

8.2.2.3.7 Transmit Flow Control Status - TFCS (0x0000CE00)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TC_XON	7:0	0xFF	RO	NA	TC is in FC XON state.
RESERVED	31:8	0x0	RSV	NA	Reserved.

8.2.2.3.8 Flow Control Configuration - FCCFG (0x00003D00)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	2:0	0x0	RSV	NA	Reserved.
TFCE	4:3	0x0	RW	NA	Transmit Flow Control Enable. These bits Indicate that the device transmits flow control packets (XON/XOFF frames) based on receive fullness. If auto negotiation is enabled, this bit should be set by software to the negotiated flow control value. 00b = Transmit flow control disabled. 01b = Link flow control enabled. 10b = Priority flow control enabled. 11b = Reserved. Note: Priority flow control should be enabled in DCB mode only.
RESERVED	31:5	0x0	RSV	NA	Reserved.

8.2.2.4 PCIe Registers

This section contains the registers used to control the PCIe core behavior.

8.2.2.4.1 PCIe Function Status 1 - PCI_STATUS1 (0x00011028)

Reset source: PCIe In-band Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FUNC_VALID	0	0b	RO	NA	Func valid - 0b - Function is disabled 1b - Function is enabled Note: This bit is valid to firmware even when the function is disabled
RESERVED	31:1	0x0	RSV	NA	

8.2.2.4.2 PCIe Control Extended Register - GCR_EXT (0x00011050)

Reset source: PCIe In-band Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VT_MODE	1:0	0x0	RW	NA	VT Mode of operation defines the allocation of physical registers to the VFs. Software must set this field the same as GPIE. VT_Mode.00 = noVT - Reserved. 01 = VT16 - Resources are allocated to 16 VFs. 10 = VT32 - Resources are allocated to 32 VFs. 11 = VT64 - Resources are allocated to 64 VFs.
RESERVED	3:2	0x0	RSV	NA	Reserved.
APBACD	4	0b	RW	NA	Auto PBA Clear Disable. When set to 1b, software can clear the PBA only by direct write to clear access to the PBA bit. When set to 0b, any active PBA entry is cleared on the falling edge of the appropriate interrupt request to the PCIe block. The appropriate interrupt request is cleared when software sets the associated interrupt mask bit in the EIMS (re-enabling the interrupt) or by direct write to clear to the PBA.
RESERVED	30:5	0x00	RSV	NA	Reserved
RESERVED	31				Reserved

8.2.2.4.3 PCIe Revision ID - PCI_REVID (0x00011098)

Reset source: PCIe In-band Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
NVM_REVID	7:0	0x00	RW	NA	Value of Rev ID loaded from NVM
RESERVED	31:8	0x00	RSV	NA	Reserved

8.2.2.4.4 PCIe Default Revision ID - PCI_DREVID (0x00011540)

Reset source: PCIe In-band Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
DEFAULT_REVID	7:0	0x00	RO	NA	Mirroring of Default Rev ID prior to NVM load
RESERVED	31:8	0x00	RSV	NA	Reserved

8.2.2.4.5 Function Requester ID Information Register - FUNC_RID (0x00011070)

Reset source: PCIe In-band Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FUNCTION_NUMBER	2:0	0x0	RO	NA	Function number assigned to the function based on BIOS/OS Enumeration.
DEVICE_NUMBER	7:3	0x0	RO	NA	Device Number Device number assigned to the function based on BIOS/OS Enumeration.
BUS_NUMBER	15:8	0x0	RO	NA	Bus Number Bus Number assigned to the function based on BIOS/OS Enumeration.
RESERVED	31:16	0x0	RSV	NA	Reserved

8.2.2.4.6 PCIe VM Pending Index - PCI_VMINDEX (0x00011530)

Reset source: PCIe In-band Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VMINDEX	8:0	0x0	RW	NA	VM Index. Software sets the VMINDEX that its transaction pending flag should be reflected in the PCI_VMPEND register. The VM index is an absolute index in the range of 0 through 63. It can be set by the software only to VMs that the PF owns and only to VMs which are not assigned to a VF.
RESERVED	31:9	0x0	RSV	NA	Reserved

8.2.2.4.7 PCIe VM Pending Status - PCI_VMPEND (0x00011538)

Reset source: PCIe In-band Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PENDING	0	0b	RO	NA	PCIe transaction pending status. The reported VM is controlled by the VMINDEX field in the PCI_VMINDEX register. This flag is set to 1b as long as there is at least one PCIe transaction, pending for its completion.
RESERVED	31:1	0x0	RSV	NA	Reserved

8.2.2.4.8 PCIe Interrupts Enable - PCI_IENA (0x00011528)

Reset source: PCIe In-band Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PCI_ERR_EN	31:0	0x0	RW	NA	Each bit corresponds to a particular error event as described in section "Proprietary Error Reporting"

8.2.2.4.9 PCIe Interrupt Cause - PCI_ICAUSE (0x00011520)

Reset source: PCIe In-band Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PCI_ERR_CAUSE	31:0	0x0	RW1C	DYNAMIC	Each bit corresponds to a particular error event as described in section "Proprietary Error Reporting"

8.2.2.4.10 PCIe Errors Reported - PCI_PCIERR (0x00011140)

Reset source: PCIe In-band Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PCI_ERR_REPORT	31:0	0x0	RO	NA	Each bit corresponds to a particular error event as described in section "Proprietary Error Reporting"

8.2.2.4.11 PCIe Statistic Control Register #1 - PCI_GSCL_1 (0x00011800)

Reset source: PCIe In-band Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GIO_COUNT_EN_0	0	0b	RW	NA	Enables PCIe* statistic counter number 0.
GIO_COUNT_EN_1	1	0b	RW	NA	Enables PCIe* statistic counter number 1.
GIO_COUNT_EN_2	2	0b	RW	NA	Enables PCIe* statistic counter number 2.
GIO_COUNT_EN_3	3	0b	RW	NA	Enables PCIe* statistic counter number 3.
LBC_ENABLE_0	4	0b	RW	NA	When set, statistics counter 0 operates in Leaky Bucket mode. When cleared, Leaky Bucket mode is disabled and the counter is incremented by one for each event.
LBC_ENABLE_1	5	0b	RW	NA	When set, statistics counter 1 operates in Leaky Bucket mode. When cleared, Leaky Bucket mode is disabled and the counter is incremented by one for each event.
LBC_ENABLE_2	6	0b	RW	NA	When set, statistics counter 2 operates in Leaky Bucket mode. When cleared, Leaky Bucket mode is disabled and the counter is incremented by one for each event.
LBC_ENABLE_3	7	0b	RW	NA	When set, statistics counter 3 operates in Leaky Bucket mode. When cleared, Leaky Bucket mode is disabled and the counter is incremented by one for each event.

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PCI_COUNT_LAT_EN	8	0b	RW	NA	PCIe Count Latency Enable - Enables the latency counter 0b - Disable the latency counter 1b - Enable the latency counter
PCI_COUNT_LAT_EV	13:9	0x00	RW	NA	PCIe Count Latency Event - Selects the event to be measured
PCI_COUNT_BW_EN	14	0b	RW	NA	PCIe Count Bandwidth Enable - Enables the bandwidth counter 0b - Disable the bandwidth counter 1b - Enable the bandwidth counter
PCI_COUNT_BW_EV	19:15	0x00	RW	NA	PCIe Count Bandwidth Event - Selects the event to be measured
RESERVED	27:20	0x0	RW	NA	Reserved
GIO_64_BIT_EN	28	0b	RW	NA	Enables two 64-bit counters instead of four 32-bit counters.
GIO_COUNT_RESET	29	0b	RWS	NA	Reset indication of PCIe* statistic counters. Write of zero has no effect.
GIO_COUNT_STOP	30	0b	RWS	NA	Stop indication of PCIe* statistic counters. Write of zero has no effect.
GIO_COUNT_START	31	0b	RWS	NA	Start indication of PCIe* statistic counters. Write of zero has no effect.

8.2.2.4.12 PCIe Statistic Control Registers #2 - PCI_GSCL_2 (0x00011804)

Reset source: PCIe In-band Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GIO_EVENT_NUM_0	7:0	0x0	RW	NA	Event number that counter 0 counts (GSCN_0).
GIO_EVENT_NUM_1	15:8	0x0	RW	NA	Event number that counter 1 counts (GSCN_1).
GIO_EVENT_NUM_2	23:16	0x0	RW	NA	Event number that counter 2 counts (GSCN_2).
GIO_EVENT_NUM_3	31:24	0x0	RW	NA	Event number that counter 3 counts (GSCN_3).

8.2.2.4.13 PCIe Statistic Counter Registers #0...#3 - PCI_GSCN_0_3[n] (0x00011820 + 0x4*n, n=0...3)

Reset source: PCIe In-band Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
EVENT_COUNTER	31:0	0x0	RO	NA	Event counter as defined in PCI_GSCL_2.GIO_EVENT_NUM fields. These registers are stuck at their maximum value of 0xFF...F.

8.2.2.4.14 PCIe Statistic Control Register #5...#8 - PCI_GSCL_5_8[n] (0x00011810 + 0x4*n, n=0...3)

Reset source: PCIe In-band Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
LBC_THRESH_OLD_N	15:0	0x00	RW	NA	Threshold for the Leaky Bucket Counter n
LBC_TIMER_N	31:16	0x00	RW	NA	Time period between decrementing the value in Leaky Bucket Counter n. The time period is defined in usec units

8.2.2.4.15 PCIe Byte Counter Low - PCI_BYTCTL (0x00011548)

Reset source: PCIe In-band Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PCI_COUNT_BW_BCT	31:0	0x00	RO	NA	This register contains the low double-word of a 64-bit counter that counts PCIe payload bytes This register gets stuck at its maximum value of 0xFF...F.

8.2.2.4.16 PCIe Byte Counter High - PCI_BYTCTH (0x00011544)

Reset source: PCIe In-band Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PCI_COUNT_BW_BCT	31:0	0x00	RO	NA	This register contains the high double-word of a 64-bit counter that counts PCIe payload bytes This register gets stuck at its maximum value of 0xFF...F.

8.2.2.4.17 PCIe LCB Address Port - PCI_LCBADD (0x00011788)

Reset source: PCIe In-band Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
ADDRESS	17:0	0x0	RW	NA	An 18-bit address register used for indirect access to PCIe core module's registers.
RESERVED	19:18	0x0	RSV	NA	Reserved
BLOCK_ID	27:20	0x0	RW	NA	An ID of a sub-unit in the PCIe unit. Supported values are: 0. NPQ: RLAN 1. NPQ: TLAN 2. NPQ: RX_PE 3. NPQ: TX_PE 4. NPQ: PMAT 5. NPQ: MNG 6. NPQ: TDPU 7. PQ: RLAN 8. PQ: TLAN 9. PQ: RX_PE 10. PQ: TX_PE 11. PQ: PMAT 12. PQ: MNG 13. PQ: RDPU 14. VDM 15. Don't care 16. HIU: CFG/Slave/MSG/MSI-X 126: LCB's internal config space registers 127: LCB's internal memory space registers
RESERVED	31:28	0x0	RSV	NA	Reserved

8.2.2.4.18 PCIe LCB Data Port - PCI_LCBDATA (0x00011734)

Reset source: PCIe In-band Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
LCB_DATA	31:0	0x0	RW	NA	A 32-bit data register used for indirect access to PCIe core module's register. The address is taken from the PCI_LCBADD register

8.2.2.4.19 PCIe Firmware Control - PCI_FWCTRL (0x00011040)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TCO_ISOLATE	0	0b	RO	NA	Set by Firmware to block all host write access if instructed to do so by BMC.
RESERVED	31:1	0x0	RSV	NA	Reserved

8.2.2.4.20 PCIe CSR Access Timeout - PCI_CSRT0 (0x00011044)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
CSR_TO	15:0	0x000F	RO	NA	Defines the timeout value in micro second for CSR accesses
MSIX_TO	31:16	0x0028	RO	NA	Defines the timeout value in micro second for MSI-x table accesses

8.2.2.4.21 PCI Flash Access Timeout - PCI_FLASHTO (0x00011054)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PCI_FLASHTO	31:0	0x00000FA0	RO	NA	Defines the timeout value in micro second for flash accesses

8.2.2.5 PCIe Configuration Space Setting Registers

This section contains registers used to set the default setting of the PCIe configuration space. These registers are reset and loaded from the NVM at PCIe reset.

8.2.2.5.1 PCIe Global Config 2 - PCI_CNF2 (0x000110F8)

Reset source: PCIe Immediate In-band Reset

NVM Load: Type1/2 in PCIe Immediate In-band Reset Register Autoload module, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	0	0b	RSV	NA	Reserved
CACHELINE_SIZE	1	0b	RW	NA	Cache Line Size - Determines the system cache line size 0b - 64B 1b - 128B This field is loaded from NVM
MSI_X_PF_N	12:2	0x3F	RW	NA	MSI_X PF Table Size - System software reads this field to determine the MSI-X Table Size N, which is encoded as N-1. This field is loaded from the NVM MSI_X_N_PF field and reflects the same field in the PCIe MSI-X configuration (Message Control Register)
MSI_X_VF_N	23:13	0x02	RW	NA	MSI_X VF Table Size - System software reads this field to determine the MSI-X Table Size N for VFs, which is encoded as N-1. This field is loaded from the NVM MSI_X_N_VF field and reflects the same field in the PCIe MSI-X configuration (VF MSI-X Control Register)
NUM_VFS	31:24	0x40	RW	NA	The number of VFs requested by the function. Valid values are 0 - 64. If value is zero, then SR-IOV capability is hidden.

8.2.2.5.2 PCI BAR Control - PCI_LBARCTRL (0x00011048)

Reset source: PCIe Immediate In-band Reset

NVM Load: Type1/2 in PCIe Immediate In-band Reset Register Autoload module, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PREFBAR	0	1b	RW	NA	Prefetchable bit indication in the memory BARs (should be set when 64bit BARs are used) 0b - BARs are marked as non prefetchable 1b - BARs are marked as prefetchable
BAR32	1	0b	RW	NA	BAR 32bit Enable. 0b - 64bit BAR addressing mode is selected. 1b - 32bit BARs are enabled.
CSRSIZE	2	0b	RO	NA	The CSRSIZE and FL_BAR_SIZE fields define the usable FLASH size and CSR mapping window size.
RESERVED	5:3	0x0	RSV	NA	Reserved
FL_BAR_SIZE	8:6	0x4	RW	NA	This field indicates the size of the external Flash as = 64KB x (2 ** FL_BAR_SIZE) as used to define exposure in the BAR. The following values are supported: 100b - 1MB 101b - 2MB Else - Reserved

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	10:9	0x0	RSV	NA	Reserved
EXROM_BAR_SIZE	13:11	0x3	RW	NA	This field indicates the size of the expansion ROM BAR as = 64KB x (2 ** EXROM_BAR_SIZE) 000b corresponds to a 64KB size 111b corresponds to a 8MB size Default value is 512KB
RESERVED	31:14	0x0	RSV	NA	Reserved

8.2.2.5.3 PCIe Global Config - PCI_GLBL_CNF (0x000110B8)

Reset source: LAN Power Good Reset

NVM Load: Type1/2 in LAN Power Good Reset Register Autoload module, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	1:0	0x0	RSV	NA	Reserved
WAKE_PIN_EN	2	0b	RW	NA	When set to 1b enables the use of PE_WAKE_N pin for PME event in all power states
PCIE_CLKGATE_DIS	3	1b	RW	NA	When set to 0b enables dynamic clock gating of the PCIE clocks (LCB, HIU & CSR).
PCIE_CLKGATE_L1_ONLY	4	1b	RW	NA	When set to 1b, and if PCIE_CLKGATE_DIS is 0, the clock gating of the PCIE clocks will be only when the PCIE is in L1 state.
PCIE_PCLK_GATE_EN	5	0b	RW	NA	When set to 1b, and if PCIE_CLKGATE_DIS is 0, the PCIE PCLK will also be dynamically gated.
RESERVED	7:6	0x0	RSV	NA	Reserved
PCIE_CLKGATE_TIMER	15:8	0x10	RW	NA	Clock gating idle timer. This field defines the number of clocks (CSR clock) of idle-detect before gating the PCIE clocks. Note: It is not recommended to set a value smaller than 0x10.
RESERVED	31:19	0x0	RSV	NA	Reserved

8.2.2.5.4 PCIe Capabilities Support - PCI_CAPSUP (0x00011088)

Reset source: PCIe Immediate In-band Reset

NVM Load: Type1/2 in PCIe Immediate In-band Reset Register Autoload module, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PCIE_VER	0	1b	RW	NA	Determines the PCIe capability version 0b = Capability version: 0x1. 1b = Capability version: 0x2.
DOE_EN	1	0b	RW	NA	Enable support of Data Object Exchange (DOE) Extended Capability
LTR_EN	2	0b	RW	NA	A value of 1b indicates support for the PCIe Latency Tolerance Reporting (LTR) Capability
TPH_EN	3	1b	RW	NA	A value of 1b indicates support for the PCIe TPH Requester Capability
ARI_EN	4	1b	RW	NA	A value of 1b indicates support for the PCIe ARI Capability
IOV_EN	5	1b	RW	NA	A value of 1b indicates support for the PCIe SR-IOV Capability
ACS_EN	6	1b	RW	NA	A value of 1b indicates support for the PCIe ACS Capability

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
SEC_EN	7	1b	RW	NA	A value of 1b indicates support for the Secondary PCI Express Extended Capability
RESERVED	8	0b	RSV	NA	reserved
DLFE_EN	9	0b	RW	NA	A value of 1b indicates support for PCIe Data Link Feature capability A value of 1b indicates support for PCIe Data Link Feature capability
GEN4_EXT_EN	10	1b	RW	NA	A value of 1b indicates support for PCIe physical layer 13.0 GT/s extended capability
GEN4_MARG_EN	11	1b	RW	NA	A value of 1b indicates support for PCIe lane margining at receiver capability
RESERVED	13:12	0x0	RSV	NA	reserved
SNPS_RAS_EN	14	1b	RW	NA	Enable support of Synopsys VSEC RAS (DES and DP) Capabilities
ECRC_MCTP_GEN	15	0b	RW	NA	ECRC Generation for MCTP: 0 = Do not add ECRC to MCTP packets even if ECRC is enabled. 1 = Add ECRC to MCTP packets if ECRC is enabled via the ECRC Generation Enable field in PCIe Advanced Error Capabilities and Control Register.
ECRC_GEN_EN	16	0b	RW	NA	Loaded into the ECRC Generation Capable bit of the PCIe Configuration registers.
ECRC_CHK_EN	17	0b	RW	NA	Loaded into the ECRC Check Capable bit of the PCIe Configuration registers.
IDO_EN	18	1b	RW	NA	Enables ID-based ordering (IDO)
MSI_MASK	19	1b	RW	NA	MSI per-vector masking setting. This bit is loaded to the masking bit (bit 8) in the Message Control of the MSI Configuration Capability structure.
CSR_CONF_EN	20	1b	RW	NA	Enables Access to CSRs via the PCI Configuration Space. See Section "Configuration Access to Internal Registers and Memories"
RESERVED	29:21	0x0	RSV	NA	Reserved
LOAD_SUBSYS_ID	30	0b	RW	NA	Load Subsystem IDs - When set to 1b, indicates that the device loads its PCIe subsystem ID and sub-system vendor ID from NVM
LOAD_DEV_ID	31	0b	RW	NA	Load Device ID - When set to 1b, indicates that the device loads its PCI device IDs from NVM

8.2.2.5.5 PCIe Link Capabilities - PCI_LINKCAP (0x00011090)

Reset source: PCIe Immediate In-band Reset

NVM Load: Type1/2 in PCIe Immediate In-band Reset Register Autoload module, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	5:0				Reserved
MAX_PAYLOAD	8:6	0x2	RW	NA	Max Payload Size Supported - Loaded to the PCIe Device Capabilities Register Supported values are 000b to 100b (128B to 2KB). Otherwise, keep HW default
MAX_LINK_WIDTH	12:9	0x07	RW	NA	Max Link Width - Loaded to the PCIe Link Capabilities Register Values: 0001b - Limit max link width to x1 0011b - Limit max link width to x4 0111b - Do not limit max link width. negotiate to the max width supported by the link Else - Reserved
RESERVED	31:13	0x0	RSV	NA	reserved

8.2.2.5.6 PCIe Capabilities Control - PCI_CAPCTRL (0x00011080)

Reset source: PCIe Immediate In-band Reset

NVM Load: Type1/2 in PCIe Immediate In-band Reset Register Autoload module, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VPD_EN	0	0b	RW	NA	0b - The PCIe VPD Capability is not present and is not exposed 1b - The PCIe VPD Capability is present and exposed
RESERVED	31:1	0x0	RSV	NA	Reserved

8.2.2.5.7 PCIe Serial Number MAC Address Low - PCI_SERL (0x000110F0)

Reset source: PCIe Immediate In-band Reset

NVM Load: Type1/2 in PCIe Immediate In-band Reset Register Autoload module, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
SER_NUM_L	31:0	0x0	RW	NA	The low DW of the Ethernet MAC address used to generate the PCIe serial number

8.2.2.5.8 PCIe Serial Number MAC Address High - PCI_SERH (0x00011078)

Reset source: PCIe Immediate In-band Reset

NVM Load: Type1/2 in PCIe Immediate In-band Reset Register Autoload module, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
SER_NUM_H	15:0	0x0	RW	NA	The high Word of the Ethernet MAC address used to generate the PCIe serial number
RESERVED	31:16	0x0	RSV	NA	Reserved

8.2.2.5.9 PCIe PF Configuration - PCI_CNF (0x00011000)

Reset source: PCIe Immediate In-band Reset

NVM Load: Type1/2 in PCIe Immediate In-band Reset Register Autoload module, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	2:0	0x0	RSV	NA	Reserved
EXROM_DIS	3	1b	RW	NA	Expansion ROM Disable 0b - The Expansion ROM BAR in the PCI configuration space is enabled 1b - The Expansion ROM BAR in the PCI configuration space is disabled
IO_BAR	4	0b	RW	NA	I/O BAR support 0b - I/O BAR is not supported 1b - I/O BAR is supported
INT_PIN	6:5	0x0	RW	NA	Controls the value advertised in the Interrupt Pin field of the PCI configuration header for this function. Values of 00b, 01b, 10b and 11b correspond to INTA#, INTB#, INTC# and INTD# respectively. The value advertised in the PCI configuration header is the value loaded from NVM + 1. The default value for port 1 is 0x1
RESERVED	31:7	0x0	RSV	NA	Reserved

8.2.2.5.10 PCIe Storage Class - PCI_CLASS (0x00011038)

Reset source: PCIe Immediate In-band Reset

NVM Load: Type1/2 in PCIe Immediate In-band Reset Register Autoload module, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
STORAGE_CLASS	0	0b	RW	NA	0b - The class code of this port is set to 0x020000 (LAN) 1b - The class code of this port is set to 0x010000 (SCSI)
RESERVED	31:1	0x0	RSV	NA	Reserved

8.2.2.5.11 PCIe PM Support - PCI_PMSUP (0x000110A0)

Reset source: PCIe Immediate In-band Reset

NVM Load: Type1/2 in PCIe Immediate In-band Reset Register Autoload module, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
ASPM_SUP	1:0	0x3	RW	NA	Active State Link PM Support is loaded to the PCIe Link Capabilities register.
L0S_EXIT_LAT	4:2	0x5	RW	NA	L0s Exit Latency. Loaded to the L0s Exit Latency field in the PCIe Link Capabilities register.
L1_EXIT_LAT	7:5	0x4	RW	NA	L1 Exit Latency. Loaded to L1 Exit Latency field in the PCIe Link Capabilities register.
L0S_ACC_LAT	10:8	0x3	RW	NA	Loaded to the Endpoint L0s Acceptable Latency field in the PCIe Device Capabilities register.
L1_ACC_LAT	13:11	0x6	RW	NA	Loaded to the Endpoint L1 Acceptable Latency field in the PCIe Device Capabilities register.
SLOT_CLK	14	1b	RW	NA	Slot Clock Configuration. Loaded to the PCIe Link Status register.

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
OBFF_SUP	16:15	0x0	RW	NA	Loaded to the OBFF Supported field in the PCIe Device Capabilities 2 register. Must be set to 0b in the NVM (OBFF is not supported).
RESERVED	31:17	0x0	RSV	NA	Reserved.

8.2.2.5.12 PCIe Power Data Register - PCI_PWRDATA (0x00011060)

Reset source: PCIe Immediate In-band Reset

NVM Load: Type1/2 in PCIe Immediate In-band Reset Register Autoload module, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
D0_POWER	7:0	0x0	RW	NA	The value in this field is reflected in the PCI Power Management Data register of the LAN functions for D0 power consumption and dissipation (Data_Select = 0 or 4).
COMM_POWER	15:8	0x0	RW	NA	The value in this field is reflected in the PCI Power Management Data register of function 0 when the Data_Select field is set to 8 (common function).
D3_POWER	23:16	0x0	RW	NA	The value in this field is reflected in the PCI Power Management Data register of the LAN functions for D3 power consumption and dissipation (Data_Select = 3 or 7).
DATA_SCALE	25:24	0x1	RO	NA	The value in this field is reflecting the PCI Power Management Data register of the LAN functions for Data Scale. This field indicates the scaling factor that's used when interpreting the value of the Data register. The normal value is 01b (indicating 0.1 watt/units). Reserved (00b) for any other values of Data_Select.
RESERVED	31:26	0x0	RSV	NA	Reserved

8.2.2.5.13 PCIe Vendor ID - PCI_VENDORID (0x0001103C)

Reset source: PCIe In-band Reset

NVM Load: Type1/2 in PCIe In-band Reset Register Autoload module, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VENDOR_ID	15:0	0x8086	RW	NA	Contains the Vendor ID exposed in offset 0x0 in the config space of all functions. A value of 0xFFFF is ignored.
RESERVED	31:16	0x0	RSV	NA	

8.2.2.5.14 PCIe PF Device ID - PCI_PFDEVID (0x00011008)

Reset source: PCIe Immediate In-band Reset

NVM Load: Type1/2 in PCIe Immediate In-band Reset Register Autoload module, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PF_DEV_ID_LAN	15:0	0x57AC	RW	NA	Contains the Device ID for this PF when the function has an Ethernet device Class Code
PF_DEV_ID_SAN	31:16	0x57AC	RW	NA	Contains the Device ID for this PF when the function has a SCSI device Class Code

8.2.2.5.15 PCIe VF Device ID - PCI_VFDEVID (0x00011010)

Reset source: PCIe Immediate In-band Reset

NVM Load: Type1/2 in PCIe Immediate In-band Reset Register Autoload module, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VF_DEV_ID_LAN	15:0	0x57AD	RW	NA	Contains the device ID for this PF's VFs when the function has an Ethernet device class code.
VF_DEV_ID_SAN	31:16	0x57AD	RW	NA	Contains the device ID for this PF's VFs when the function has a SCSI device class code.

8.2.2.5.16 PCIe Upper Address - PCI_UPADD (0x000110E8)

Reset source: PCIe Immediate In-band Reset

NVM Load: Type1/2 in PCIe Immediate In-band Reset Register Autoload module, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	0	0b	RSV	NA	Reserved
ADDRESS	31:1	0x0	RW	NA	Address - Bits [31:1] correspond to bits [63:33] in the PCIe address space, respectively. Bits that should not be set in the addresses on the PCIe should be set in this field. For example in a 48 bit architecture, the value of this field should be 0xFFFF0000

8.2.2.6 Interrupt Registers

8.2.2.6.1 Extended Interrupt Cause Register - EICR (0x00000800)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RTXQ	15:0	0x0	RW1C	DYNAMIC	Receive/Transmit Queue Interrupts. One bit per queue or a bundle of queues, activated on receive/transmit events. The mapping of queue to the RTXQ bits is done by the IVAR registers.
FLOW_DIRECTOR	16	0b	RW1C	DYNAMIC	Flow director exception is activated by one of the following events: 1. Filter removal failed (there was no matched filter to be removed). 2. The number of remaining free filters in the flexible filter table exceeds (goes below) the FDIRCTRL.Full-Thresh. 3. Filter programming failed due to no space in the flow director table. Note: This case should not happen if the driver handles the FDIRCTRL.Full-Thresh event.
RX_MISS	17	0b	RW1C	DYNAMIC	Missed packet interrupt is activated for each received packet that overflows the Rx packet buffer (overflow). Note: The packet is dropped and also increments the associated RXMPC[n] counter.
PCI_EXCEPTION	18	0b	RW1C	DYNAMIC	The PCI exception is activated by one of the events described in Proprietary Error Reporting section. The specific PCI event is error reported in the PCI_ICAUSE register:
MAILBOX	19	0b	RW1C	DYNAMIC	VF to PF mailbox interrupt. Caused by a VF write access to the PF mailbox or by a malicious event detection
LSC	20	0b	RW1C	DYNAMIC	Link Status Change. This bit is set each time the link status changes (either from up to down, or from down to up).
MNG_GP	21	0b	RW1C	DYNAMIC	General Purpose Interrupt can be trigger by FW through AUX register.
MNG	22	0b	RW1C	DYNAMIC	Manageability Event Detected. Indicates that a manageability event happened. When the device is in power-down mode, the BMC might generate a PME for the same events that would cause an interrupt when the device is at the D0 state. This interrupt bit is also used to alert the host when the BMC IP address was changed or when EEC.MAC_DONE bit is set by firmware.
TS	23	0b	RW1C	DYNAMIC	Thermal Sensor event. Indicates that a thermal sensor trip point was crossed.
TIMESYNC	24	0b	RW1C	DYNAMIC	TimeSync Interrupt. Indicates that a Time Sync event has occurred. Check TSIM register for precise cause.
GPI_SDP0	25	0b	RW1C	DYNAMIC	General Purpose Interrupt on SDP0. If GPI interrupt detection is enabled on this pin (via GPIE), this interrupt cause is set when a transition to high is sampled on SDP0.
GPI_SDP1	26	0b	RW1C	DYNAMIC	General Purpose Interrupt on SDP1. If GPI interrupt detection is enabled on this pin (via GPIE), this interrupt cause is set when a transition to high is sampled on SDP1.

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GPI_SDP2	27	0b	RW1C	DYNAMIC	General Purpose Interrupt on SDP2. If GPI interrupt detection is enabled on this pin (via GPIE), this interrupt cause is set when the SDP2 is sampled high.
ECC	28	0b	RW1C	DYNAMIC	Unrecoverable ECC error. This bit is set when one of the following occurs: - An unrecoverable error is detected in one of the device memories. - A fatal error occurred in the PHY device. Software should issue a software reset following this error.
PHY_GLOBAL_INTERRUPT	29	0b	RW1C	DYNAMIC	PHY Interrupt (Non Fatal).
TCP_TIMER	30	0b	RW1C	DYNAMIC	TCP Timer Expired. This bit is set when the timer expires.
OTHER_CAUSE	31	0b	ROS	NA	Other Cause Interrupt. Activated when any bit (29:16) in the Extended Interrupt Cause Register (EICR) is set and its relevant mask bit in the EIMS is enabled.

8.2.2.6.2 Extended Interrupt Cause Set Register - EICS (0x00000808)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
INTERRUPT_CAUSE_SET	30:0	0x0	WO	NA	Setting any bit in this field, sets its corresponding bit in the EICR and generates an interrupt if enabled by EIMS register.
RESERVED	31	0b	RSV	NA	Reserved.

8.2.2.6.3 Extended Interrupt Mask Set/Read Register - EIMS (0x00000880)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
INTERRUPT_ENABLE	30:0	0x0	RWS	NA	Each bit that is set to 1b enables its corresponding interrupt in the EICR. Writing a 1b to any bit sets it. Writing 0b has no impact. Reading this register provides a map of those interrupts that are enabled.
RESERVED	31	0b	RSV	NA	Reserved.

8.2.2.6.4 Extended Interrupt Mask Clear Register - EIMC (0x00000888)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
INTERRUPT_MASK	30:0	0x0	WO	NA	Writing a 1b to any bit clears its corresponding bit in the EIMS disabling the corresponding interrupt in the EICR. Writing 0b has no impact. Reading this register provides no meaningful data.
RESERVED	31	0b	RSV	NA	Reserved.

8.2.2.6.5 Extended Interrupt Auto Clear Register - EIAC (0x00000810)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RTXQ_AUTO_CLEAR	15:0	0x0	RW	NA	At 1b each bit enables auto clear of the corresponding RTXQ bits in the EICR following interrupt assertion. At 0b the corresponding bits in the EICR are not auto cleared.
RESERVED	29:16	0x0	RSV	NA	Reserved.
TCP_TIMER_AUTO_CLEAR	30	0b	RW	NA	At '1b, this bit enables auto clear of the TCP timer interrupt cause in the EICR following interrupt assertion. At 0b auto clear is not enabled.
RESERVED	31	0b	RSV	NA	Reserved.

Note: Bits 29:16 should never be set to auto clear since they share the same MSI-X vector.

8.2.2.6.6 Extended Interrupt Auto Mask Enable register - EIAM (0x00000890)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
AUTO_MASK	30:0	0x0	RW	NA	At 1b each bit enables auto set and clear of its corresponding bits in the EIMS. Note: If any of the Auto Mask enable bits is set, the GPIE.EIAME bit must be set as well.
RESERVED	31	0b	RSV	NA	Reserved.

8.2.2.6.7 Extended Interrupt Cause Set Registers 1 - EICS1 (0x00000A90)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
INTERRUPT_CAUSE_SET	31:0	0x0	WO	NA	Setting any bit in these registers, sets its corresponding bit in the EICR[n] and generates an interrupt if enabled by EIMS[n] register. Reading this register provides no meaningful data.

8.2.2.6.8 Extended Interrupt Cause Set Registers 2 - EICS2 (0x00000A94)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
INTERRUPT_CAUSE_SET	31:0	0x0	WO	NA	Setting any bit in these registers, sets its corresponding bit in the EICR[n] and generates an interrupt if enabled by EIMS[n] register. Reading this register provides no meaningful data.

8.2.2.6.9 Extended Interrupt Mask Set/Read Registers 1 - EIMS1 (0x0000AA0)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
INTERRUPT_ENABLE	31:0	0x0	RWS	NA	Each bit at 1b enables its corresponding interrupt in the EICR[n]. Writing 1b to any bit sets it. Writing 0b has no impact. Reading this register provides a map of those interrupts that are enabled. Bits 15:0 of EIMS1 are mirrored in EIMS bits 15:0.

8.2.2.6.10 Extended Interrupt Mask Set/Read Registers 2 - EIMS2 (0x0000AA4)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
INTERRUPT_ENABLE	31:0	0x0	RWS	NA	Each bit at 1b enables its corresponding interrupt in the EICR[n]. Writing 1b to any bit sets it. Writing 0b has no impact. Reading this register provides a map of those interrupts that are enabled. Bits 15:0 of EIMS1 are mirrored in EIMS bits 15:0.

8.2.2.6.11 Extended Interrupt Mask Clear Registers 1 - EIMC1 (0x0000AB0)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
INTERRUPT_MASK	31:0	0x0	WO	NA	Writing 1b to any bit clears its corresponding bit in the EIMS[n] disabling the corresponding interrupt in the EICR[n]. Writing 0b has no impact. Reading this register provides no meaningful data.

8.2.2.6.12 Extended Interrupt Mask Clear Registers 2 - EIMC2 (0x0000AB4)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
INTERRUPT_MASK	31:0	0x0	WO	NA	Writing 1b to any bit clears its corresponding bit in the EIMS[n] disabling the corresponding interrupt in the EICR[n]. Writing 0b has no impact. Reading this register provides no meaningful data.

8.2.2.6.13 Extended Interrupt Auto Mask Enable registers 1 - EIAM1 (0x0000AD0)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
AUTO_MASK	31:0	0x0	RW	NA	At 1b each bit enables auto set and clear of its corresponding bits in the EIMS[n]. Bits 15:0 of EIAM1 are mirrored in EIAM bits 15:0. Note: If any of the Auto Mask enable bits is set, the GPIE.EIAME bit must be set as well.

8.2.2.6.14 Extended Interrupt Auto Mask Enable registers 2 - EIAM2 (0x0000AD4)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
AUTO_MASK	31:0	0x0	RW	NA	At 1b each bit enables auto set and clear of its corresponding bits in the EIMS[n]. Bits 15:0 of EIAM1 are mirrored in EIAM bits 15:0. Note: If any of the Auto Mask enable bits is set, the GPIE.EIAME bit must be set as well.

8.2.2.6.15 MSIX to EITR Select - EITRSEL (0x0000894)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VFSELECT	31:0	0x0	RW	NA	Each bit 'n' in this register selects the VF index (32+'n') or PF interrupt source for the EITR registers (VF 0-31 are not multiplexed as described in Section 7.3.4.3.3). At 0x0 it selects the PF and at 0x1 it selects the VF.

8.2.2.6.16 Extended Interrupt Throttle Registers - EITR[n] (0x00012300 + 0x4*n, n=0...104 and 0x00000820 + 0x4*(n-105), n=105...128)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	2:0	0x0	RSV	NA	Reserved.
ITR_INTERVAL	11:3	0x0	RW	NA	Minimum inter-interrupt interval specified in 2.048 us units at 1 GbE and 10 GbE link. At 100 Mb/s link the interval is specified in 20.48 us units. At 0x0 interrupt throttling is disabled while any event causes an immediate interrupt. Should not be set to 0x1 or 0x2 (2/4 us).
RESERVED	15:12	0x0	RSV	NA	Reserved.
LLI_CREDIT	20:16	0x0	RW	NA	Reflects the current credits for associated interrupt. When CNT_WDIS is not set on write cycle, this field must be set to zero.

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
ITR_COUNTERR	27:21	0x0	RW	NA	This field represents the 7 MSbits (out of 9 bits) of the ITR counter. It is a down counter that is loaded with ITR Interval value each time the associated interrupt is asserted. When the ITR counter reaches zero it stops counting and triggers an interrupt. On a write cycle, software must set this field to 0 if CNT_WDIS in this register is cleared (write enable to the ITR counter).
RESERVED	30:28	0x0	RSV	NA	Reserved.
CNT_WDIS	31	0b	RW	NA	Write disable to the LLI credit and ITR counter. When set, the LLI credit and ITR counter are not overwritten by the write access. When cleared software must set the LLI credit and ITR counter to zero, which enables an immediate interrupt on packet reception. This bit is write only and always read as zero.

Note: Additional address(es): 0x012300 + 4*(n-24), n=24...128

8.2.2.6.17 Interrupt Vector Allocation Registers - IVAR[n] (0x00000900 + 0x4*n, n=0...63)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
INT_ALLOC_0	5:0	X	RW	NA	The interrupt allocation for Rx queue ('2xN' for IVAR register 'N').
RESERVED	6	0b	RSV	NA	Reserved.
INT_ALLOC_VAL_0	7	0b	RW	NA	Interrupt Allocation Valid indication for INT_Alloc[0].
INT_ALLOC_1	13:8	X	RW	NA	The interrupt allocation for Tx queue ('2xN' for IVAR register 'N').
RESERVED	14	0b	RSV	NA	Reserved.
INT_ALLOC_VAL_1	15	0b	RW	NA	Interrupt Allocation Valid indication for INT_Alloc[1].
INT_ALLOC_2	21:16	X	RW	NA	The interrupt allocation for Rx queue ('2xN'+1 for IVAR register 'N').
RESERVED	22	0b	RSV	NA	Reserved.
INT_ALLOC_VAL_2	23	0b	RW	NA	Interrupt Allocation Valid indication for INT_Alloc[2].
INT_ALLOC_3	29:24	X	RW	NA	The interrupt allocation for Tx queue ('2xN'+1 for IVAR register 'N').
RESERVED	30	0b	RSV	NA	Reserved.
INT_ALLOC_VAL_3	31	0b	RW	NA	Interrupt Allocation Valid indication for INT_Alloc[3].

8.2.2.6.18 Miscellaneous Interrupt Vector Allocation - IVAR_MISC (0x00000A00)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
INT_ALLOC_0	6:0	X	RW	NA	Defines the MSI-X vector assigned to the TCP timer interrupt cause. The value must be in the 0-63 range.
INT_ALLOC_VAL_0	7	0b	RW	NA	Valid bit for INT_Alloc[0].
INT_ALLOC_1	14:8	X	RW	NA	Defines the MSI-X vector assigned to the "Other" interrupt cause. The value must be in the 0-63 range.
INT_ALLOC_VAL_1	15	1b	RW	NA	Valid bit for INT_Alloc[1].
RESERVED	31:16	0x0	RSV	NA	Reserved.

Note: The INT_ALLOC_VAL[1] bit default value is one to enable legacy driver functionality.

8.2.2.6.19 General Purpose Interrupt Enable - GPIE (0x00000898)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	0	0b	RSV	NA	Reserved.
SDP0_GPIEN	1	0b	RW	NA	General Purpose Interrupt Detection Enable for SDP0. If software-controllable IO pin SDP0 is configured as an input, this bit (when 1) enables use for GPI interrupt detection. Resulting in setting EICR[25] bit
SDP1_GPIEN	2	0b	RW	NA	General Purpose Interrupt Detection Enable for SDP1. If software-controllable IO pin SDP1 is configured as an input, this bit (when 1) enables use for GPI interrupt detection. Resulting in setting EICR[26] bit
SDP2_GPIEN	3	0b	RW	NA	General Purpose Interrupt Detection Enable for SDP2. If software-controllable IO pin SDP2 is configured as an input, this bit (when 1) enables use for GPI interrupt detection. Resulting in setting EICR[27] bit
MULTIPLE_MSIX	4	0b	RW	NA	MSI-X Mode. Selects between MSI-X interrupts and other interrupt modes. 0 - Legacy and MSI mode (non-MSI-X mode) 1 - MSI-X mode
OCD	5	0b	RW	NA	Other clear Disable. When set indicates that only bits 16...29 of the EICR are cleared on read. When cleared, the whole EICR is cleared on read.
EIMEN	6	0b	RW	NA	EICS immediate interrupt enable. When set, setting bit in the EICS causes a Low Latency Interrupt. If not set, the EICS interrupt waits for EITR expiration.
RESERVED	13:7				Reserved

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VT_MODE	15:14	0x0	RW	NA	Specify the number of active Virtual functions. Software must set this field the same as GCR_Ext.VT_Mode. 00: Non IOV mode 01: 16 VF mode 10: 32 VF mode 11: 64 VF mode
RESERVED	29:16	0x0	RSV	NA	Reserved.
EIAME	30	0b	RW	NA	Extended Interrupt Auto Mask enable: When set the EIMS register can be auto-cleared (depending on EIAM setting) upon interrupt assertion. In any case, the EIMS register can be auto-cleared (depending on EIAM setting) following a write-to-clear (or read) to the EICR register. Software might set the EIAME only in MSI-X mode.
PBA_SUPPORT	31	0b	RW	NA	PBA Support: When set, setting one of the extended interrupts masks via EIMS causes the PBA bit of the associated MSI-X vector to be cleared. Otherwise, the device behaves in a way supporting legacy INT-x interrupts. Note: Should be cleared when working in INT-x or MSI mode and set in MSI-X mode.

8.2.2.7 MSI-X Table Registers

See description of the “MSI-X Capability” section within the “PCIe Programming Interface” chapter.

8.2.2.7.1 MSI-X PBA Clear - PBACL[n] (0x000110C0 + 0x4*n, n=0...1)

Reset source: PCIe In-band Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PENBITCLR	31:0	0x0	RW	NA	MSI-X Pending bits Clear. Writing 1b to any bit clears it's content; writing 0b has no effect. Reading this register returns the MSIPBA.PENBIT value.

8.2.2.7.2 VF MSI-X PBA Clear - VFPBACL[n] (0x000110C8 + 0x4*n, n=0...5)

Reset source: PCIe In-band Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PENBITCLR	31:0	0x0	RW	NA	MSI-X Pending bits Clear. Writing 1b to any bit clears it's content; writing 0b has no effect. Reading this register returns the MSIPBA.PENBIT value.

Notes:

- These registers reflects the VFPBACL bits of the VFs.
- The PBA is a vector of 192 bits. The vector starts at bit 31 of register n=5 and ends at bit 0 of register n=0 (reverse ordering). Each VF has 3 bits in this vector while, PENBIT[2...0] of VF=m are mapped to bits $m * 3 \dots m * 3 + 2$. Explicitly, PENBIT[2] of VF0 is at bit 31 of register n=5 and so on.

8.2.2.8 Receive Registers

8.2.2.8.1 Filter Control Register - FCTRL (0x00005080)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	0	0b	RSV	NA	Reserved.
SBP	1	0b	RW	NA	Store bad packets. 0 = do not store; 1 = store. Notes: - CRC errors before the SFD are ignored. Any packet must have a valid SFD (RX_DV with no RX_ER in the XGMII/GMII i/f) in order to be recognized by the device (even bad packets). - Packets with errors are not routed to manageability even if this bit is set. - Erroneous packets might be routed to the default queue rather than the originally intended queue. - In packets shorted than 64 bytes the checksum errors might be hidden while MAC errors are reported. - Packet with Valid Error (caused by Byte Error or Illegal Error) might have data contamination in the last 8 bytes when stored in the Host memory if the store-bad-packet bit is set.
RESERVED	6:2	0x0	RSV	NA	Reserved.
TPE	7	0b	RW	NA	Tag Promiscuous Enable When set, and PFVTCTL.POOLING_MODE= 01b, any packet with E-tag is accepted. If cleared and PFVTCTL.POOLING_MODE= 01b packets with E-tag are accepted according to RAH/RAL registers. If PFVTCTL.POOLING_MODE != 01b, this bit is ignored.
MPE	8	0b	RW	NA	Multicast promiscuous enable. 0 = disabled; 1 = enabled. When set, all received multicast and broadcast packets pass L2 filtering and can be directed to the MNG or Host by a one of the decision filters.
UPE	9	0b	RW	NA	Unicast promiscuous enable. 0 = disabled; 1 = enabled.
BAM	10	0b	RW	NA	Broadcast Accept Mode. 0 - ignore broadcast packets to Host 1 - accept broadcast packets to Host
RESERVED	31:11	0x0	RSV	NA	Reserved.

Note: Before receive filters are updated/modified the RXCTRL.RXEN bit should be set to zero. After the proper filters have been set the RXCTRL.RXEN bit can be set to 1 to re-enable the receiver.

8.2.2.8.2 VLAN Control Register - VLNCTRL (0x00005088)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VET	15:0	0x8100	RW	NA	VLAN Ether Type (the VLAN Tag Protocol Identifier - TPID). This register contains the type field hardware that's matched against to recognize an 802.1Q (VLAN) Ethernet packet. For proper operation, software must not change the default setting of this field (802.3ac standard defines it as 0x8100). This field must be set to the same value as the VT field in the DMATXCTL register. Note: This field appears in little endian order (the upper byte is first on the wire (VLNCTRL.VET[15:8])).
RESERVED	26:16	0x0	RSV	NA	Reserved.
UP_FIRST_TAG_EN	27	1b	RW	NA	If set, the UP used for traffic class determination is taken from the first tag with UP, otherwise, it is taken from the inner VLAN.
DEI	28	0b	RW	NA	Drop Eligible Indicator Bit Value. If DEIEN is set to 1b, then .1q packets with DEI equal to this field are accepted; otherwise, the .1q packet is discarded.
DEIEN	29	0b	RW	NA	Drop Eligible Indicator Enable. 0b = Disabled (DEI bit not compared to decide packet acceptance). 1b = Enabled (DEI from packet must match the DEI field to accept a .1q packet).
VFE	30	0b	RW	NA	VLAN Filter Enable. 0b = Disabled (VFTA filter table does not decide packet acceptance). 1b = Enabled (VFTA filter table decides packet acceptance for .1q packets).
RESERVED	31	0b	RSV	NA	Reserved.

8.2.2.8.3 Multicast Control Register - MCSTCTRL (0x00005090)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MO	1:0	0x0	RW	NA	Multicast Offset. This determines which bits of the incoming multicast address are used in looking up the bit vector. 00 = [47:36]; 01 = [46:35]; 10 = [45:34]; 11 = [43:32].
MFE	2	0b	RW	NA	Multicast Filter Enable 0 - the multicast table array filter (MTA[n]) is disabled 1 - the multicast table array (MTA[n]) is enabled
RESERVED	31:3	0x0	RSV	NA	Reserved.

8.2.2.8.4 Packet Split Receive Type Register - PSRTYPE[n] (0x0000EA00 + 0x4*n, n=0...63)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PSR_TYPE0	0	0b	RW	NA	Reserved.
PSR_TYPE1	1	0b	RW	NA	Split received NFS packets after NFS header.
PSR_TYPE2	2	0b	RW	NA	Reserved.
PSR_TYPE3	3	0b	RW	NA	Reserved.
PSR_TYPE4	4	0b	RW	NA	Split received TCP packets after TCP header.
PSR_TYPE5	5	0b	RW	NA	Split received UDP packets after UDP header.
PSR_TYPE6	6	0b	RW	NA	Reserved.
PSR_TYPE7	7	0b	RW	NA	Reserved.
PSR_TYPE8	8	0b	RW	NA	Split received IPv4 packets after IPv4 header.
PSR_TYPE9	9	0b	RW	NA	Split received IPv6 packets after IPv6 header.
PSR_TYPE10	10	0b	RW	NA	Reserved.
PSR_TYPE11	11	0b	RW	NA	Reserved.
PSR_TYPE12	12	0b	RW	NA	Split received L2 packets after L2 header.
PSR_TYPE13	13	0b	RW	NA	Reserved.
PSR_TYPE14	14	0b	RW	NA	Reserved.
PSR_TYPE15	15	0b	RW	NA	Split on tunnel header
PSR_TYPE16	16	0b	RW	NA	Split on outer L2 header
PSR_TYPE17	17	0b	RW	NA	Reserved.
PSR_TYPE18	18	0b	RW	NA	Reserved.
RESERVED	28:19	X	RSV	NA	Reserved.
RQPL	31:29	X	RW	NA	Defines the number of bits to use for RSS redirection of packets dedicated to this pool. Valid values are zero, 0001b and 0010b. The default value should be 0010b, meaning that up to 4 queues can be enabled for this pool. A value of 0001b means that up to 2 queues can be enabled for this pool. A value of zero means that all the traffic of the pool is sent to queue 0 of the pool. This field is used only if MRQC.MRQE equals 1001b, 1010b, 1011b, 1110b or 1111b

Note: Additional address(es): 0x05480 + 4*n, n=0... 15

8.2.2.8.5 Receive Checksum Control - RXCSUM (0x00005000)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	9:0	0x00	RSV	NA	Reserved.
ICMPV6XSUM	10	1b	RW	NA	ICMPv6 Checksum Enable. 0b = Disable ICMPv6 checksum calculation. 1b = Enable ICMPv6 checksum calculation. Note: ICMPv6 checksum offload is supported only for packets sent to firmware for Proxying.
RESERVED	11	0b	RSV	NA	Reserved

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
IPPCSE	12	0b	RW	NA	IP Payload Checksum Enable
PCSD	13	0b	RW	NA	RSS/Fragment Checksum status selection. The Fragment Checksum and IP Identification fields are mutually exclusive with the RSS hash. Only one of the two options is reported in the Rx descriptor. When set to 1, the extended descriptor write-back has the RSS field. When set to 0, it contains the fragment checksum.
RESERVED	31:14	0x0	RSV	NA	Reserved.

Note: This register should only be initialized (written) when the receiver is not enabled (for example, only write this register when RXCTRL.RXEN = 0).

8.2.2.8.6 Receive Filter Control Register - RFCTL (0x00005008)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	4:0	0x0	RSV	NA	Reserved.
RESERVED	5				Reserved
NFSW_DIS	6	0b	RW	NA	NFS Write disable. Disable filtering of NFS write request headers.
NFSR_DIS	7	0b	RW	NA	NFS Read disable. Disable filtering of NFS read reply headers.
NFS_VER	9:8	0x0	RW	NA	NFS Version recognized by hardware. 00 NFS version 2 01 NFS version 3 10 NFS version 4 11 Reserved. for future use
IPV6_DIS	10	0b	RW	NA	IPv6 disable. Disable IPv6 packet filtering Internal use only - should not be set to 1.
IP6XSUM_DIS	11	0b	RW	NA	IPv6 Xsum disable Disable XSUM on IPv6 packets Internal use only - should not be set to 1.
RESERVED	13:12	0x0	RSV	NA	Reserved.
IPFRSP_DIS	14	0b	RW	NA	IP Fragment Split Disable When this bit is set the header of IP fragmented packets are not set. Internal use only. Should not be set to 1b.
RESERVED	15	0b	RSV	NA	Reserved.
IPV6_EXDIS	16	0b	RW	NA	IPv6 Extension Header Disable, Chicken bit to disable the IPv6 extension headers parsing for XSUM offload, Header split and Filtering: 0 - parse and recognize allowed IPV6 extension headers (Hop-by-Hop, Destination Options, and Routing) 1 - do not recognize above extension headers Internal use only - should not be set to 1.
RESERVED	31:17	0x0	RSV	NA	Reserved. Should be written with 0 to ensure future compatibility.

8.2.2.8.7 Multicast Table Array - MTA[n] (0x00005200 + 0x4*n, n=0...127)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
BIT_VECTOR	31:0	X	RW	NA	Word wide bit vector specifying 32 bits in the multicast address filter table. This device provides Multicast filtering for 4096 Multicast Addresses by providing single bit entry per Multicast Address. Those 4096 address locations organized in Multicast Table Array 128 registers of 32 bits each one. Only 12 bits out of 48 bit Destination Address are considered as Multicast Address. Those 12 bits can be selected by MO field of MCSTCTRL register. The 7 MS bits of the Ethernet MAC Address (out of the 12 bits) selects the register index while the 5 LS bits (out of the 12 bits) selects the bit within a register.

8.2.2.8.8 Receive Address Low - RAL[n] (0x0000A200 + 0x8*n, n=0...127)

Reset source: Master Reset, NVM Load: Dedicated, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RAL	31:0	X	RW	NA	Receive Address Low. The lower 32 bits of the 48 bit Ethernet MAC Address. Note: Field is defined in Big Endian (LS byte of RAL is first on the wire).

Note: The first 16 MAC addresses are also mapped to CSR addresses 0x5400 - 0x5478 for backward compatibility.

8.2.2.8.9 Receive Address High - RAH[n] (0x0000A204 + 0x8*n, n=0...127)

Reset source: Master Reset, NVM Load: Dedicated, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RAH	15:0	X	RW	NA	Receive Address High. The upper 16 bits of the 48 bit Ethernet MAC Address. Note: Field is defined in Big Endian (MS byte of RAH is Last on the wire).
RESERVED	29:16	0x0	RSV	NA	Reserved.
ADTYPE	30	0b	RW	NA	Address Type: 0: MAC 1: E-tag Fixed to zero in RAH[0].
AV	31	0b	RW	NA	Address valid. All Receive Addresses are not initialized by hardware and software should init them before receive is enabled. If the NVM is present, the Receive Address[0] is loaded from the NVM and its Address Valid field is set to 1b after a software or PCI reset or NVM read.

Notes:

- The first receive address register (RAR0) is also used for exact match pause frame checking (DA matches the first register). RAR0 should always be used to store the individual Ethernet MAC address of the adapter. After reset, if the NVM is present, the first register (Receive Address Register 0) is loaded from the IA field in the NVM, its Address Select field is 00b, and its Address Valid field is 1b. If no NVM is present, the Address Valid field is 0b. The Address Valid field for all of the other registers are 0b.
- The first 16 MAC addresses are also mapped to CSR addresses 0x5404 - 0x547C for backward compatibility.

8.2.2.8.10 Receive Address Low Alias - RAL_ALIAS[n] (0x00005400 + 0x8*n, n=0...15; RW)

Reset source: Master Reset, NVM Load: Dedicated, Scope: PF

Fields definitions are the same as defined on [Section 8.2.2.8.8](#)

8.2.2.8.11 Receive Address High Alias - RAH_ALIAS[n] (0x00005404 + 0x8*n, n=0...15; RW)

Reset source: Master Reset, NVM Load: Dedicated, Scope: PF

Fields definitions are the same as defined on [Section 8.2.2.8.9](#)

8.2.2.8.12 MAC Pool Select Array - MPSAR[n] (0x0000A600 + 0x4*n, n=0...255)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
POOL_ENA	31:0	X	RW	NA	Pool Enable bit array. Each couple of registers '2*n' and '2*n+1' are associated with Ethernet MAC Address Filter 'n' as defined by RAL[n] and RAH[n]. Each bit when set, enables packet reception with the associated Pools as described below: Bit 'i' in register '2*n' is associated with POOL 'i'. Bit 'i' in register '2*n+1' is associated with POOL '32+i'.

8.2.2.8.13 VLAN Filter Table Array - VFTA[n] (0x0000A000 + 0x4*n, n=0...127)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VLAN_FLT	31:0	X	RW	NA	VLAN Filter. Each bit 'i' in register 'n' affects packets with VLAN VID equal to 32*n+i. 128 VLAN Filter registers compose a table of 4096 bits that cover all possible VIDs. Each bit when set, allows packets with the associated VID to pass. Each bit when cleared, blocks packets with this VID.

8.2.2.8.14 Multiple Receive Queues Command Register - MRQC (0x0000EC80)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MRQE	3:0	0x0	RW	NA	Multiple Receive Queues Enable - defines allocation of the Rx queues per RSS, virtualization and DCB. 0000b - RSS disabled 0001b - RSS only - Single set of RSS 64 queues 0010b - DCB and RSS disabled - 8 TCs, each allocated 1 queue 0011b - DCB and RSS disabled - 4 TCs, each allocated 1 queue 0100b - DCB and RSS - 8 TCs, each allocated 16 RSS queues 0101b - DCB and RSS - 4 TCs, each allocated 32 RSS queues 0110b - Reserved. 0111b - Reserved. 1000b - Virtualization only - 64 pools, no RSS, each pool allocated 2 queues 1001b - Virtualization and RSS - 62 pools, each allocated 2 RSS queues - 1 pool allocated 4 queues. 1010b - Virtualization and RSS - 32 pools, each allocated 4 RSS queues 1011b - Virtualization and RSS - 64 pools, each allocated 2 RSS queues 1100b - Virtualization and DCB - 16 pools, each allocated 8 TCs 1101b - Virtualization and DCB - 32 pools, each allocated 4 TCs 1110b - Virtualization, DCB and RSS - 16 pools, each allocated 4 TCs, each allocated 2 RSS queues. 1111b - Virtualization and RSS - 60 pools, each allocated 2 RSS queues - 2 pools, each allocated 4 queues.
RESERVED	12:4	0x0	RSV	NA	Reserved.
MULTIPLE_RS S	13	0b	RW	NA	Multiple RSS Enable When set to 1b the device uses up to 64 RSS Keys (one per pool) each RSS has a redirection table with 64 entry of 2 bits each. When set to 0b the device uses a single RSS Key (Legacy) This bit is relevant only if MRQC.MRQE = 1001b, 1010b, 1011b, 1110b, or 1111b
RESERVED	15:14				Reserved

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RSS_FIELD_ENABLE	31:16	0x0	RW	NA	Each bit, when set, enables a specific field selection to be used by the hash function. Several bits can be set at the same time. Bit[16] - Enable TcpIPv4 hash function Bit[17] - Enable IPv4 hash function Bit[18] - Reserved. Bit[19] - Reserved. Bit[20] - Enable IPv6 hash function Bit[21] - Enable TcpIPv6 hash function Bit[22] - Enable UdpIPv4 Bit[23] - Enable UdpIPv6 Bit[24] - Reserved. Bits[31:25] - Reserved. Zero Note: On Tunnel packets IPv4-IPv6 only the IPv4 header might be used for the RSS filtering

8.2.2.8.15 Multiple Receive Queues Command Register Alias - MRQC_ALIAS (0x00005818; RW)

Reset source: Master Reset, NVM Load: No, Scope: PF

Fields definitions are the same as defined on [Section 8.2.2.8.14](#)

8.2.2.8.16 RSS queues per Traffic class register - RQTC (0x0000EC70)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RQTC0	2:0	0x4	RW	NA	Defines the number of bits to use for RSS redirection of packets dedicated to traffic class 0. A value of zero means that all the traffic of TC0 is sent to queue 0 of the traffic class. This field is used only if MRQC.Multiple Receive Queues Enable = 0100b or 0101b. A value of 7 in this field is not valid and is will be considered as if zero is written.
RESERVED	3	0b	RSV	NA	Reserved.
RQTC1	6:4	0x4	RW	NA	Defines the number of bits to use for RSS redirection of packets dedicated to traffic class 1. A value of zero means that all the traffic of TC1 is sent to queue 0 of the traffic class. This field is used only if MRQC.Multiple Receive Queues Enable = 0100b or 0101b. A value of 7 in this field is not valid and is will be considered as if zero is written.
RESERVED	7	0b	RSV	NA	Reserved.
RQTC2	10:8	0x4	RW	NA	Defines the number of bits to use for RSS redirection of packets dedicated to traffic class 2. A value of zero means that all the traffic of TC2 is sent to queue 0 of the traffic class. This field is used only if MRQC.Multiple Receive Queues Enable = 0100b or 0101b. A value of 7 in this field is not valid and is will be considered as if zero is written.
RESERVED	11	0b	RSV	NA	Reserved.

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RQTC3	14:12	0x4	RW	NA	Defines the number of bits to use for RSS redirection of packets dedicated to traffic class 3. A value of zero means that all the traffic of TC3 is sent to queue 0 of the traffic class. This field is used only if MRQC.Multiple Receive Queues Enable = 0100b or 0101b. A value of 7 in this field is not valid and is will be considered as if zero is written.
RESERVED	15	0b	RSV	NA	Reserved.
RQTC4	18:16	0x4	RW	NA	Defines the number of bits to use for RSS redirection of packets dedicated to traffic class 4. A value of zero means that all the traffic of TC4 is sent to queue 0 of the traffic class. This field is used only if MRQC.Multiple Receive Queues Enable = 0100b. A value of 7 in this field is not valid and is will be considered as if zero is written.
RESERVED	19	0b	RSV	NA	Reserved.
RQTC5	22:20	0x4	RW	NA	Defines the number of bits to use for RSS redirection of packets dedicated to traffic class 5. A value of zero means that all the traffic of TC5 is sent to queue 0 of the traffic class. This field is used only if MRQC.Multiple Receive Queues Enable = 0100b. A value of 7 in this field is not valid and is will be considered as if zero is written.
RESERVED	23	0b	RSV	NA	Reserved.
RQTC6	26:24	0x4	RW	NA	Defines the number of bits to use for RSS redirection of packets dedicated to traffic class 6. A value of zero means that all the traffic of TC6 is sent to queue 0 of the traffic class. This field is used only if MRQC.Multiple Receive Queues Enable = 0100b. A value of 7 in this field is not valid and is will be considered as if zero is written.
RESERVED	27	0b	RSV	NA	Reserved.
RQTC7	30:28	0x4	RW	NA	Defines the number of bits to use for RSS redirection of packets dedicated to traffic class 7. A value of zero means that all the traffic of TC7 is sent to queue 0 of the traffic class. This field is used only if MRQC.Multiple Receive Queues Enable = 0100b. A value of 7 in this field is not valid and is will be considered as if zero is written.
RESERVED	31	0b	RSV	NA	Reserved.

8.2.2.8.17 RSS Random Key Register - RSSRK[n] (0x0000EB80 + 0x4*n, n=0...9)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
K0	7:0	0x0	RW	NA	RSS Key Byte '4*n+0' of the RSS random key, for each register 'n'.
K1	15:8	0x0	RW	NA	RSS Key Byte '4*n+1' of the RSS random key, for each register 'n'.
K2	23:16	0x0	RW	NA	RSS Key Byte '4*n+2' of the RSS random key, for each register 'n'.
K3	31:24	0x0	RW	NA	RSS Key Byte '4*n+3' of the RSS random key, for each register 'n'.

Note: Additional address(es): 0x05C80 + 4*n, n=0...9

8.2.2.8.18 Redirection Table - RETA[n] (0x0000EB00 + 0x4*n, n=0...31)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
ENTRY0	5:0	X	RW	NA	Entry0 defines the RSS output index for hash value '4*4*n+0'. While 'n' is the register index, equals to 0...31.
RESERVED	7:6	0x0	RSV	NA	Reserved.
ENTRY1	13:8	X	RW	NA	Entry1 defines the RSS output index for hash value '4*4*n+1'. While 'n' is the register index, equals to 0...31.
RESERVED	15:14	0x0	RSV	NA	Reserved.
ENTRY2	21:16	X	RW	NA	Entry2 defines the RSS output index for hash value '4*4*n+2'. While 'n' is the register index, equals to 0...31.
RESERVED	23:22	0x0	RSV	NA	Reserved.
ENTRY3	29:24	X	RW	NA	Entry3 defines the RSS output index for hash value '4*4*n+3'. While 'n' is the register index, equals to 0...31.
RESERVED	31:30	0x0	RSV	NA	Reserved.

Note: The contents of the redirection table are not defined following reset of the Memory Configuration Registers. System software must initialize the Table prior to enabling multiple receive queues. It might also update the redirection table during run time. Such updates of the table are not synchronized with the arrival time of received packets. Therefore, it is not guaranteed that a table update takes effect on a specific packet boundary.

8.2.2.8.19 Extended Redirection Table - ERETA[n] (0x0000EE80 + 0x4*n, n=0...95)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
ENTRY0	5:0	X	RW	NA	Entry0 defines the RSS output index for hash value '4*[n+32]'. While 'n' is the register index, equals to 0...95.
RESERVED	7:6	0x0	RSV	NA	Reserved.
ENTRY1	13:8	X	RW	NA	Entry1 defines the RSS output index for hash value '4*[n+32]+1'. While 'n' is the register index, equals to 0...95.
RESERVED	15:14	0x0	RSV	NA	Reserved.
ENTRY2	21:16	X	RW	NA	Entry2 defines the RSS output index for hash value '4*[n+32]+2'. While 'n' is the register index, equals to 0...95.
RESERVED	23:22	0x0	RSV	NA	Reserved.
ENTRY3	29:24	X	RW	NA	Entry3 defines the RSS output index for hash value '4*[n+32]+3'. While 'n' is the register index, equals to 0...95.
RESERVED	31:30	0x0	RSV	NA	Reserved.

Note: The contents of the extended redirection table are not defined following reset of the Memory Configuration Registers. System software must initialize the Table prior to enabling multiple receive queues. It might also update the redirection table during run time. Such updates of the table are not synchronized with the arrival time of received packets. Therefore, it is not guaranteed that a table update takes effect on a specific packet boundary.

8.2.2.8.20 Per Pool RSS Random Key Register - VFRSSRK[n,m] (0x00018000 + 0x4*n + 0x40*m, n=0...15, m=0...63)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
K0	7:0	0x0	RW	NA	RSS Key Byte '4*n+0' of the RSS random key, for each register 'n'.
K1	15:8	0x0	RW	NA	RSS Key Byte '4*n+1' of the RSS random key, for each register 'n'.
K2	23:16	0x0	RW	NA	RSS Key Byte '4*n+2' of the RSS random key, for each register 'n'.
K3	31:24	0x0	RW	NA	RSS Key Byte '4*n+3' of the RSS random key, for each register 'n'.

Note: Only the 10 first registers in each VF array are implemented (0x00 - 0x24)

8.2.2.8.21 Per Pool Redirection Table - VFRETA[n,m] (0x00019000 + 0x4*n + 0x40*m, n=0...15, m=0...63)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
ENTRY0	1:0	X	RW	NA	Entry0 defines the RSS output index for hash value '4*4*n+0'. While 'n' is the register index, equals to 0...31.
RESERVED	7:2	0x0	RSV	NA	Reserved.
ENTRY1	9:8	X	RW	NA	Entry1 defines the RSS output index for hash value '4*4*n+1'. While 'n' is the register index, equals to 0...31.
RESERVED	15:10	0x0	RSV	NA	Reserved.
ENTRY2	17:16	X	RW	NA	Entry2 defines the RSS output index for hash value '4*4*n+2'. While 'n' is the register index, equals to 0...31.
RESERVED	23:18	0x0	RSV	NA	Reserved.
ENTRY3	25:24	X	RW	NA	Entry3 defines the RSS output index for hash value '4*4*n+3'. While 'n' is the register index, equals to 0...31.
RESERVED	31:26	0x0	RSV	NA	Reserved.

Notes:

- The content of the redirection tables is not defined following reset of the Memory Configuration registers.
- System software must initialize the table prior to enabling multiple receive queues. It might also update the redirection table during run time. Such updates of the table are not synchronized with the arrival time of received packets. Therefore, it is not guaranteed that a table update takes effect on a specific packet boundary.

8.2.2.8.22 SYN Packet Queue Filter - SYNQF (0x0000EC30)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
QUEUE_ENABLE	0	0b	RW	NA	When set, enables routing of Rx packets to the queue indicated in this register.
RX_QUEUE	7:1	0x00	RW	NA	Identifies an Rx queue associated with SYN packets.
RESERVED	8				Reserved
RESERVED	31:9	0x0	RSV	NA	Reserved.

8.2.2.8.23 EType Queue Filter - ETQF[n] (0x00005128 + 0x4*n, n=0...7)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
ETYPE	15:0	0x0000	RW	NA	Identifies the protocol running on top of IEEE 802. Used to route Rx packets containing this EtherType to a specific Rx queue. Note: Field is defined in Little Endian (MS byte is first on the wire).
RESERVED	19:16				Reserved
POOL	25:20	0x00	RW	NA	In virtualization modes, determines the target pool for the packet.
POOL_ENABLE	26	0b	RW	NA	In virtualization modes, enables the Pool field.
RESERVED	27				Reserved
CNM	28	0b	RW	NA	When set, a packet with this EType is identified as a CNM packet.
TX_ANTI_SPOOF	29	0b	RSV	NA	If set, this EtherType is candidate for anti spoof action on transmitted packets. The actual action applied is set according to the PFVFSPOOF.ETHERTYPEAS or PFVFSPOOF.ETHERTYPELB
IEEE_1588_TIMESTAMP	30	0b	RW	NA	When set, packets with this EType are time stamped according to the IEEE 1588 specification.
FILTER_ENABLE	31	0b	RW	NA	0b - the filter is disabled for any functionality. 1b - the filter is enabled. Exact actions are determined by separate bits.

8.2.2.8.24 EType Queue Select - ETQS[n] (0x0000EC00 + 0x4*n, n=0...7)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	15:0	0x0	RSV	NA	Reserved.
RX_QUEUE	22:16	0x0	RW	NA	Identifies the Rx queue associated with this EType.
RESERVED	30:23	0x0	RSV	NA	Reserved. f
QUEUE_ENABLE	31	0b	RW	NA	When set, enables queuing of Rx packets by the EType defined in the matching ETQF register to the queue indicated in this register.

8.2.2.8.25 E-tag EtherType Register - ETAG_ETYPE (0x00005084)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
ETAG_ETYPE	15:0	0x893F	RW	NA	Etag EtherType Value
RESERVED	30:16	0x0	RSV	NA	Reserved
VALID	31	0b	RW	NA	Valid Bit 0b = Entry is disabled. 1b = Entry is enabled.

8.2.2.8.26 VXLAN control - VXLANCTRL (0x0000507C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
UDPPORT	15:0	0x0	RW	NA	Defines the UDP port used to identify VXLAN traffic.
GENEVE_UDP_PORT	31:16	0x0	RW	NA	Defines the UDP port used to identify Geneve traffic.

8.2.2.9 Receive DMA Registers

8.2.2.9.1 Receive Descriptor Base Address Low - RDBAL[n] (0x0000D000 + 0x40*n, n=0...63 and 0x00001000 + 0x40*(n-64), n=64...127)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
ZERO	6:0	0x0	RW	NA	Ignored on writes. Returns 0 on reads.
RDBAL	31:7	X	RW	NA	Receive Descriptor Base Address Low

8.2.2.9.2 Receive Descriptor Base Address High - RDBAH[n] (0x0000D004 + 0x40*n, n=0...63 and 0x00001004 + 0x40*(n-64), n=64...127)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RDBAH	31:0	X	RW	NA	Receive Descriptor Base Address [63:32]

8.2.2.9.3 Receive Descriptor Length - RDLEN[n] (0x0000D008 + 0x40*n, n=0...63 and 0x00001008 + 0x40*(n-64), n=64...127)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
LEN	19:0	0x0	RW	NA	Descriptor Ring Length. This register sets the number of bytes allocated for descriptors in the circular descriptor buffer. It must be 128B aligned (7 LS bit must be set to zero). Note: Validated Lengths up to 128K (8K descriptors).
RESERVED	31:20	0x0	RSV	NA	Reads as 0. Should be written to 0 for future compatibility.

Note: Additional address(es): 0x0D008 + 0x40*(n-64), n=64...127

8.2.2.9.4 Receive Descriptor Head - RDH[n] (0x0000D010 + 0x40*n, n=0...63 and 0x00001010 + 0x40*(n-64), n=64...127)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RDH	15:0	0x0	RO	NA	Receive Descriptor Head. This register holds the head pointer for the receive descriptor buffer in descriptor units (16B datum). The RDH is controlled by hardware.
RESERVED	31:16	0x0	RSV	NA	Reserved. Should be written with 0.

Note: Additional address(es): 0x0D010 + 0x40*(n-64), n=64...127

8.2.2.9.5 Receive Descriptor Tail - RDT[n] (0x0000D018 + 0x40*n, n=0...63 and 0x00001018 + 0x40*(n-64), n=64...127)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RDT	15:0	0x0	RW	NA	Receive Descriptor Tail
RESERVED	31:16	0x0	RSV	NA	Reads as 0. Should be written to 0 for future compatibility.

Notes:

- Software should write an even number to the tail register, if the Packet Split feature is used.
- The tail pointer should be set to one descriptor beyond the last empty descriptor in Host Descriptor Ring.

8.2.2.9.6 Receive Descriptor Control - RXDCTL[n] (0x00001028 + 0x40*n, n=0...63 and 0x0000D028 + 0x40*(n-64), n=64...127)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RLPML	13:0	0x0	RW	NA	Long packet size filter defined in bytes. Packets larger than the RLPML are discarded. The filter is enabled by the RLPML_EN bit in this register. This field might not be supported per Rx queue in future products. Note: The packet size compared to RLPML includes any optional timestamp added into the packet.
RESERVED	14	0b	RSV	NA	Reserved. (software might Read and Write maintaining backward compatibility.)
RLPML_EN	15	0b	RW	NA	Enable Long packet size filter ('1' = enable).
RESERVED	22:16	0x00	RSV	NA	Reserved. (software might Read and Write maintaining backward compatibility.)
RESERVED	24:23	0x0	RSV	NA	Reserved.
ENABLE	25	0b	RW	NA	Receive Queue enable. When set, the ENABLE bit enables the operation of the specific receive queue, upon read - get the actual status of the queue (internal indication that the queue is actually enabled/disabled).
RESERVED	29:26	0x00	RSV	NA	Reserved.
VME	30	0b	RW	NA	VLAN Mode Enable 1 = Strip VLAN tag from received 802.1Q packets destined to this queue. 0 = Do not strip VLAN tag.
RESERVED	31	0b	RSV	NA	Reserved

Note: Additional address(es): 0x0D028 + 0x40*(n-64), n=64...127

8.2.2.9.7 Split Receive Control Registers - SRRCTL[n] (0x00001014 + 0x40*n, n=0...63 and 0x0000D014 + 0x40*(n-64), n=64...127)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
BSIZEPACKET	4:0	0x2	RW	NA	Receive Buffer Size for Packet Buffer. The value is in 1K byte resolution. Value can be from 1K bytes to 16K bytes. Default buffer size is 2K. This field should not be set to 0.
RESERVED	7:5	0x0	RSV	NA	Reserved. Should be written with 0 to ensure future compatibility.
BSIZEHEADER	13:8	0x4	RW	NA	Receive Buffer Size for Header Buffer. The value is in 64 bytes resolution. Value can be from 64 bytes to 1024 bytes. Note: The max supported header size is limited to 1023. Default buffer size is 256 bytes. This field must be greater than 0 if the value of DESCTYPE is greater or equal to 2. Values above 1024 bytes are reserved for internal use only.
RESERVED	21:14	0x0	RSV	NA	Reserved.
RDMTS	24:22	0x0	RW	NA	Receive Descriptor Minimum Threshold Size. An interrupt associated with this queue is asserted whenever the number of free descriptors is decreased to RDMTS * 64 (this event is considered as Rx ring buffer "almost empty").
DESCTYPE	27:25	0x0	RW	NA	Define the descriptor type in Rx 000: Legacy 001: Advanced descriptor one buffer 010: Advanced descriptor header splitting 011: Reserved. 100: Reserved. 101: Advanced descriptor header splitting always use header buffer. 110- 111: reserved
DROP_EN	28	0b	RW	NA	Drop Enabled: If set to 1, packets received to the queue when no descriptors are available to store them are dropped.
RESERVED	31:29	0x0	RSV	NA	Reserved. Should be written with 0 to ensure future compatibility.

Notes:

- Additional address(es): 0x0D014 + 0x40*(n-64), n=64...127 / 0x02100 + 4*n, [n=0...15]
- BSIZEHEADER must be bigger than 0 if DESCTYPE is equal to 010, 011 100 or 101.

8.2.2.9.8 Split Receive Control Registers alias - SRRCTL_ALIAS[n] (0x00002100 + 0x4*n, n=0...15; RW)

Reset source: Master Reset, NVM Load: No, Scope: PF

Fields definitions are the same as defined on [Section 8.2.2.9.7](#)

8.2.2.9.9 Receive DMA Control Register - RDRXCTL (0x00002F00)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	1:0	00b			Reserved
PSP	2	0b	RW	NA	Pad Small Receive packets. If this field is set, in virtualized operating mode, strip CRC (HLREG0.RXCRCSTRP) should be set.
DMAIDONE	3	0b	RO	NA	DMA Init Done. When read as 1 indicates that the DMA init cycle is done (RO).
RESERVED	7:4	0x0	RSV	NA	Reserved.
TPH_AUTOLEARN	8	0b	RSV	NA	TPH hints auto learn If set, the device learns the CPUID value in this register from the completer ID in read completion of receive descriptors fetch. Note: If TPH is disabled, this bit should not be set, as the learn capability is independent of the TPH capability
RESERVED	27:9	0x0	RSV	NA	Reserved
MBINTEN	28	0b	RW	NA	Malicious Behavior interrupt enable for DMA Rx errors
MDP_EN	29	0b	RW	NA	Malicious driver protection enable for DMA Rx 0b = mechanism is disabled. 1b = mechanism is enabled.
RESERVED	31:30	0x0	RSV	NA	Reserved.

8.2.2.9.10 Receive Packet Buffer Size - RXPBSIZE[n] (0x00003C00 + 0x4*n, n=0...7)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	9:0	0x0	RSV	NA	Reserved.
SIZE	19:10	0x180	RW	NA	Receive Packet Buffer Size for traffic class 'n' while 'n' is the register index. The size is defined in KB units and the default size of PB[0] is 384 KB. The default size of PB[1-7] is also 384 KB but it is meaningless in non-DCB mode. When DCB mode is enabled the size of PB[1-7] must be set to meaningful values. The total meaningful allocated PB sizes plus the size allocated to the flow director filters should be equal to 384 KB. Possible PB allocation in DCB mode for 8 x TCs is 0x30 (48 KB) for all PBs. Other possible setting of 4 x TCs is 0x60 (96 KB) for all PB[0-3] and 0x0 for PB[4-7]. See section 3.7.5.3.5 (Packet Buffer Size) for other optional settings with/without the flow director filters. Note: In any setting the RXPBSIZE[0] must always be enabled (greater than zero). Default value is 0x180 (384 KB).
RESERVED	31:20	0x0	RSV	NA	Reserved.

8.2.2.9.11 Receive Control Register - RXCTRL (0x00003000)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RXEN	0	0b	RW	NA	Receive Enable. When set to 0, packets are not received.
RESERVED	31:1	0x0	RSV	NA	Reserved.

8.2.2.10 Transmit Registers

8.2.2.10.1 DMA Tx TCP Max Allow Size Requests - DTXMXSZRQ (0x00008100)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MAX_BYTES_NUM_REQ	11:0	0x10	RW	NA	Max allowed number of bytes requests. The maximum allowed amount of 256 bytes outstanding requests. If the total size request is higher than the amount in the field no arbitration is done and no new packet is requested.
RESERVED	31:12	0x0	RSV	NA	Reserved.

8.2.2.10.2 DMA Tx Control - DMATXCTL (0x00004A80)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TE	0	0b	RW	NA	Transmit Enable. When set, this bit enables the transmit operation of the DMA-Tx.
RESERVED	2:1	00b	RSV	NA	Reserved
GDV	3	0b	RW	NA	Global Double VLAN mode. When set, this bit enables the Double VLAN mode. Should be set to the same value as CTRL_EXT.EXTENDED_VLAN
RESERVED	4				Reserved
MDP_EN	5	0b	RW	NA	Malicious driver protection enable for DMA Tx 0b = mechanism is disabled. 1b = mechanism is enabled.
MBINTEN	6	0b	RW	NA	Malicious Behavior interrupt enable for DMA Tx errors
TPH_AUTOLEARN	7	0b	RW	NA	TPH hints auto learn If set, the device learns the CPUID value in this register from the completer ID in read completion of transmit descriptors fetch. Note: If TPH is disabled, this bit should not be set, as the learn capability is independent of the TPH capability
RESERVED	15:8	0x0	RSV	NA	Reserved
VT	31:16	0x8100	RW	NA	VLAN Ether-Type (the VLAN Tag Protocol Identifier - TPID). For proper operation, software must not change the default setting of this field (802.3ac standard defines it as 0x8100). This field must be set to the same value as the VET field in the VLNCTRL register.

8.2.2.10.3 DMA Tx TCP Flags Control Low - DTXTCPFLGL (0x00004A88)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TCP_FLG_FIR ST_SEG	11:0	0xFF6	RW	NA	TCP Flags first segment. Bits that make AND operation with the TCP flags at TCP header in the first segment.
RESERVED	15:12	0x00	RSV	NA	Reserved.
TCP_FLG_MI D_SEG	27:16	0xFF6	RW	NA	TCP Flags middle segments. The low bits that make AND operation with the TCP flags at TCP header in the middle segments.
RESERVED	31:28	0x00	RSV	NA	Reserved.

8.2.2.10.4 DMA Tx TCP Flags Control High - DTXTCPFLGH (0x00004A8C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TCP_FLG_LST _SEG	11:0	0xF7F	RW	NA	TCP Flags last segment. Bits that make AND operation with the TCP flags at TCP header in the last segment.
RESERVED	31:12	0x00	RSV	NA	Reserved.

8.2.2.10.5 Transmit Descriptor Base Address Low - TDBAL[n] (0x00006000 + 0x40*n, n=0...127)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
ZERO	6:0	0x0	RW	NA	Ignored on writes. Returns 0 on reads.
TDBAL	31:7	X	RW	NA	Transmit Descriptor Base Address Low

Note: This register contains the lower bits of the 64 bit descriptor base address. The lower 7 bits are ignored. The Transmit Descriptor Base Address must point to a 128B aligned block of data.

8.2.2.10.6 Transmit Descriptor Base Address High - TDBAH[n] (0x00006004 + 0x40*n, n=0...127)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TDBAH	31:0	X	RW	NA	Transmit Descriptor Base Address [63:32]

Note: This register contains the upper 32 bits of the 64 bit Descriptor base address.

8.2.2.10.7 Transmit Descriptor Length - TDLEN[n] (0x00006008 + 0x40*n, n=0...127)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
LEN	19:0	0x0	RW	NA	Descriptor Ring Length. This register sets the number of bytes allocated for descriptors in the circular descriptor buffer. It must be 128B aligned (7 LS bit must be set to zero). Note: Validated Lengths up to 128K (8K descriptors).
RESERVED	31:20	0x0	RSV	NA	Reads as 0. Should be written to 0.

8.2.2.10.8 Transmit Descriptor Head - TDH[n] (0x00006010 + 0x40*n, n=0...127)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TDH	15:0	0x0	RO	NA	Transmit Descriptor Head
RESERVED	31:16	0x0	RSV	NA	Reserved. Should be written with 0.

Notes:

- This register contains the head pointer for the transmit descriptor ring. It points to a 16B datum. Hardware controls this pointer.
- The values in these registers might point to descriptors that are still not in the host memory. As a result, the host cannot rely on these values in order to determine which descriptor to release.
- The only time that software should write to this register is after a reset (hardware reset or CTRL.RST) and before enabling the transmit function (TXDCTL.ENABLE). If software were to write to this register while the transmit function was enabled, the on-chip descriptor buffers might be invalidated and hardware could become confused.

8.2.2.10.9 Transmit Descriptor Tail - TDT[n] (0x00006018 + 0x40*n, n=0...127)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TDT	15:0	0x0	RW	NA	Transmit Descriptor Tail.
RESERVED	31:16	0x0	RSV	NA	Reads as 0. Should be written to 0 for future compatibility.

Note: This register contains the tail pointer for the transmit descriptor ring. It points to a 16B datum. Software writes the tail pointer to add more descriptors to the transmit ready queue. Hardware attempts to transmit all packets referenced by descriptors between head and tail.

8.2.2.10.10 Transmit Descriptor Control - TXDCTL[n] (0x00006028 + 0x40*n, n=0...127)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PTHRESH	6:0	0x00	RW	NA	Pre-fetch Threshold. Controls when a pre-fetch of descriptors is considered. This threshold refers to the number of valid, unprocessed transmit descriptors are in the on-chip buffer. If this number drops below PTHRESH, the algorithm considers pre-fetching descriptors from host memory. However, this fetch does not happen unless there are at least HTHRESH valid descriptors in host memory to fetch. Note: HTHRESH should be given a non-zero value each time PTHRESH is used.
RESERVED	7	0b	RSV	NA	Reserved.
HTHRESH	14:8	0x00	RW	NA	Host Threshold.
RESERVED	15	0b	RSV	NA	Reserved.
WTHRESH	22:16	0x00	RW	NA	Write-back Threshold. Controls the write back of processed transmit descriptors. This threshold refers to the number of transmit descriptors in the on-chip buffer that are ready to be written back to host memory. In the absence of external events (explicit flushes), the write back occurs only after at least WTHRESH descriptors are available for write back. Notes: - Since the default value for a write-back threshold is 0b, descriptors are normally written back as soon as they are processed. WTHRESH must be written to a non-zero value to take advantage of the write back bursting capabilities. - When WTHRESH is set to a non-zero value, the software device driver should not set the RS bit in the Tx descriptors. When WTHRESH is set to zero, the software device driver should set the RS bit in the last Tx descriptors of every packet (if TSO is the last descriptor of the entire large send). - When head write back is enabled (TDWBAL[n].Head_WB_En = 1b), the WTHRESH must be set to zero.
RESERVED	24:23	0x00	RSV	NA	Reserved.
ENABLE	25	0b	RW	NA	Transmit Queue Enable. When set, this bit enables the operation of a specific transmit queue: The default value for all queues is 0b. Setting this bit initializes all the internal registers of a specific queue. Until then, the state of the queue is kept and can be used for debug purposes. When disabling a queue, this bit is cleared only after all activity at the queue stopped. Notes: - When setting the global Tx enable DMATXCTL.TE, the Enable bit of Tx queue zero is enabled as well. - Following a write cycle by software, this bit is set by hardware only when the queue is actually enabled. When read, gets the actual status of the queue (internal indication that the queue is actually enabled/disabled).

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
SWFLSH	26	0b	RW	NA	Transmit Software Flush. This bit enables software to trigger descriptor write-back flushing, independent of other conditions. This bit is self cleared by hardware.
RESERVED	27	0b	RSV	NA	Reserved. (was Priority).
RESERVED	28				Reserved
RESERVED	29	0b	RSV	NA	Reserved.
RESERVED	31:30	0x00	RSV	NA	Reserved.

Notes:

- This register controls the fetching and write back of transmit descriptors. The three threshold values are used to determine when descriptors are read from and written to host memory.
- When WTHRESH = 0b, only descriptors with the RS bit set are written back.
- For PTHRESH and HTHRESH recommended settings, see Section 7.2.3.4.

8.2.2.10.11 Tx Descriptor Completion Write Back Address Low - TDWBAL[n] (0x00006038 + 0x40*n, n=0...127)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
HEAD_WB_EN	0	0b	RW	NA	Head Write-Back enable At '1', Head Write-back is enabled. At '0', Head Write-back is disabled. When head_WB_en is set, Tx descriptors are not written back.
RESERVED	1				Reserved
HEADWB_LOW	31:2	0x0	RW	NA	Lowest 32 bits of the head write-back memory location (DWORD aligned). Last 2 bits of this field are ignored and are always read as 0.0, meaning that the actual address is QWORD aligned.

8.2.2.10.12 Tx Descriptor Completion Write Back Address High - TDWBAH[n] (0x0000603C + 0x40*n, n=0...127)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
HEADWB_HIGH	31:0	0x0	RW	NA	Highest 32 bits of Head Write-back memory location (for 64-bit addressing)

8.2.2.10.13 Transmit Packet Buffer Size - TXPBSIZE[n] (0x0000CC00 + 0x4*n, n=0...7)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	9:0	0x0	RSV	NA	Reserved.
SIZE	19:10	0xA0	RW	NA	Transmit Packet Buffer Size of TCn. At default (no DCB), only packet buffer 0 is enabled and TXPBSIZE values for TC 1-7 are meaningless. Other than the default configuration, the following partitions are possible when DCB is enabled. Symmetrical 8 TCs partitioning: 0x14 (20KB) for TXPBSIZE[0...7]. Symmetrical 4 TCs partitioning: 0x28 (40KB) for TXPBSIZE[0...3] and 0x0 (0KB) for TXPBSIZE[4...7]. Non-symmetrical partitioning are supported as well. In order to enable wire speed transmission, it is recommended to set the transmit packet buffers to at least 2 times MSS plus PCIe latency (approximate 1 KB) when IPsec AH is not enabled (security block is not enabled or operates in path through mode). The default value is 0xA0 (160 KB)
RESERVED	31:20	0x0	RSV	NA	Reserved.

8.2.2.10.14 Manageability Transmit TC Mapping - MNGTXMAP (0x0000CD10)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MAP	2:0	0x0	RW	NA	MAP value indicates the TC that the transmit Manageability traffic is routed to.
RESERVED	31:3	0x0	RSV	NA	Reserved.

8.2.2.10.15 Multiple Transmit Queues Command Register - MTQC (0x00008120)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
DCB_ENA	0	0b	RW	NA	DCB Enabled Mode. See functionality in the table below.
VT_ENA	1	0b	RW	NA	Virtualization Enabled Mode. When set, the device supports either 16, 32, or 64 Pools. This bit should be set the same as PFVTCTL.VT_Ena.
NUM_TC_OR_Q	3:2	0x0	RW	NA	Number of TCs or Number of Tx Queues per Pools. See functionality in the Tx Queues Assignment section
RESERVED	31:4	0x0	RSV	NA	Reserved.

Notes:

- For permitted value and functionality of: DCB_ena; VT_Ena; NUM_TC_OR_Q. For Tx queue assignment in DCB and VT modes refer to Tx Queuing Schemes Table in Tx Queues Assignment Section.
- This register can be modified only as part of the init phase.

8.2.2.10.16 Tx Packet Buffer Threshold - TXPBTHRESH[n] (0x00004950 + 0x4*n, n=0...7)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
THRESH	9:0	0x96	RW	NA	Threshold used for checking room place in the Tx packet buffer of TCn. Threshold is in KB units when the packet buffer is filled up with payload over that threshold; no more data read request is sent. Default values: 0x96 (150 KB) for TXPBSIZE0. 0x0 (0 KB) for TXPBSIZE1-7. It should be set to: Packet Buffer Size - MSS. For example, if the packet buffer size is set to 20 KB in the corresponding TXPBSIZE.SIZE and if an MSS of 9.5 KB Jumbo frames is supported for TCn, it is set to: 0xA (10 KB).
RESERVED	31:10	0x0	RSV	NA	Reserved.

8.2.2.10.17 Tags EtherTypes - TAG_ETYPE (0x00017100)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
ETAG_ETHER TYPE	15:0	0x893F	RW	NA	The EtherType to use to identify E-tags.
RESERVED	31:16				Reserved

8.2.2.11 DCB Registers

DCB registers are owned by the PF in an IOV mode.

8.2.2.11.1 DCB Receive Packet plane Control and Status - RTRPCS (0x00002430)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	0	0b	RSV	NA	Reserved.
RRM	1	0b	RW	NA	Receive Recycle Mode defines the recycle mode within a BWG. 0b = No recycle. 1b = Recycle within the BWG. It is the only supported mode when DCB is enabled.
RAC	2	0b	RW	NA	Receive Arbitration Control 0b = RR, Round Robin. 1b = WSP, Weighted Strict Priority.
RESERVED	5:3	0x0	RSV	NA	Reserved.
ARBDIS	6	0b	RW	NA	DCB Arbiter Disable. When set to 1 this bit pauses the Rx packet plane arbitration state-machine. Therefore, during nominal operation this bit should be set to 0.
RESERVED	26:7	0x0	RSV	NA	Reserved.
BWGC	27	0b	RW	NA	This bit affect the Rx WSP arbiter described in Figure 7.38. When cleared (default operation) the step "T5[BWG].Credits > 0? or no GSP in the BWG?" is taking affect. When set to 1, Do not bypass BWG credits consideration before transmit. Meaning, the second part of the condition case "or no GSP in the BWG?" is not taking place. Note: Bit RTTPCS(27) has similar affect on the Tx Packet plane Control as the BWGC on
RESERVED	31:28				Reserved

8.2.2.11.2 DCB Transmit Packet plane Control and Status - RTTPCS (0x0000CD00)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	4:0	0x0	RSV	NA	Reserved.
TPPAC	5	0b	RW	NA	Transmit Packet Plane Arbitration Control 0 -RR, Round Robin (with respect to Stop Markers) 1 - SP, Strict Priority (with respect to Stop Markers)
ARBDIS	6	0b	RW	NA	DCB Arbiter Disable. When set to 1 this bit pauses the Tx Packet plane arbitration state-machine. Therefore, during nominal operation this bit should be set to 0.
RESERVED	7	0b	RSV	NA	Reserved.
TPRM	8	0b	RW	NA	Transmit Packet plane Recycle Mode defines the recycle mode within a BWG. 0 - No recycle 1 - Recycle within the BWG

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	21:9	0x0	RSV	NA	Reserved.
ARBD	31:22	0x224	RW	NA	ARB_delay. Minimum cycles delay between packets arbitration. When RTTPCS.TPPAC is set to 1 the arbitration delay is according to ARBD, otherwise the arbitration delay is 0x0. Should be kept at default in non EEDC mode. In EEDC mode, should be set to 0x004.

8.2.2.11.3 DCB Receive User Priority to Traffic Class - RTRUP2TC (0x00003020)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
UP0MAP	2:0	0x0	RW	NA	Receive User Priority 0 to traffic class Mapping. When set to n, User Priority 0 is bound to Traffic Class n. Used for two purposes: - Define into which Rx Packet Buffer incoming traffic carrying 802.1p field set to 0 is routed. - Define according to the filling status of which Rx Packet Buffer a Priority Flow Control frame with the Timer 0 field and Class Enable Vector bit 0 set is sent.
UP1MAP	5:3	0x0	RW	NA	Receive User Priority 1 to traffic class Mapping. When set to n, User Priority 1 is bound to Traffic Class n. Used for two purposes: - Define into which Rx Packet Buffer incoming traffic carrying 802.1p field set to 1 is routed. - Define according to the filling status of which Rx Packet Buffer a Priority Flow Control frame with the Timer 1 field and Class Enable Vector bit 1 set is sent.
UP2MAP	8:6	0x0	RW	NA	Receive User Priority 2 to traffic class Mapping. When set to n, User Priority 2 is bound to Traffic Class n. Used for two purposes: - Define into which Rx Packet Buffer incoming traffic carrying 802.1p field set to 2 is routed. - Define according to the filling status of which Rx Packet Buffer a Priority Flow Control frame with the Timer 2 field and Class Enable Vector bit 2 set is sent.
UP3MAP	11:9	0x0	RW	NA	Receive User Priority 3 to traffic class Mapping. When set to n, User Priority 3 is bound to Traffic Class n. Used for two purposes: - Define into which Rx Packet Buffer incoming traffic carrying 802.1p field set to 3 is routed. - Define according to the filling status of which Rx Packet Buffer a Priority Flow Control frame with the Timer 3 field and Class Enable Vector bit 3 set is sent.

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
UP4MAP	14:12	0x0	RW	NA	Receive User Priority 4 to traffic class Mapping. When set to n, User Priority 4 is bound to Traffic Class n. Used for two purposes: - Define into which Rx Packet Buffer incoming traffic carrying 802.1p field set to 4 is routed. - Define according to the filling status of which Rx Packet Buffer a Priority Flow Control frame with the Timer 4 field and Class Enable Vector bit 4 set is sent.
UP5MAP	17:15	0x0	RW	NA	Receive User Priority 5 to traffic class Mapping. When set to n, User Priority 5 is bound to Traffic Class n. Used for two purposes: - Define into which Rx Packet Buffer incoming traffic carrying 802.1p field set to 5 is routed. - Define according to the filling status of which Rx Packet Buffer a Priority Flow Control frame with the Timer 5 field and Class Enable Vector bit 5 set is sent.
UP6MAP	20:18	0x0	RW	NA	Receive User Priority 6 to traffic class Mapping. When set to n, User Priority 6 is bound to Traffic Class n. Used for two purposes: - Define into which Rx Packet Buffer incoming traffic carrying 802.1p field set to 6 is routed. - Define according to the filling status of which Rx Packet Buffer a Priority Flow Control frame with the Timer 6 field and Class Enable Vector bit 6 set is sent.
UP7MAP	23:21	0x0	RW	NA	Receive User Priority 7 to traffic class Mapping. When set to n, User Priority 7 is bound to Traffic Class n. Used for two purposes: - Define into which Rx Packet Buffer incoming traffic carrying 802.1p field set to 7 is routed. - Define according to the filling status of which Rx Packet Buffer a Priority Flow Control frame with the Timer 7 field and Class Enable Vector bit 7 set is sent.
RESERVED	31:24	0x0	RSV	NA	Reserved.

8.2.2.11.4 DCB Transmit User Priority to Traffic Class - RTTUP2TC (0x0000C800)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
UP0MAP	2:0	0x0	RW	NA	Transmit User Priority 0 to traffic class MAPPING. When set to n, User Priority 0 is bound to Traffic Class n. Used when receiving a Priority Flow Control frame with the Timer 0 field and Class Enable Vector bit 0 set, to determine which traffic class shall be paused.
UP1MAP	5:3	0x0	RW	NA	Transmit User Priority 1 to traffic class MAPPING. When set to n, User Priority 1 is bound to Traffic Class n. Used when receiving a Priority Flow Control frame with the Timer 1 field and Class Enable Vector bit 1 set, to determine which traffic class shall be paused.
UP2MAP	8:6	0x0	RW	NA	Transmit User Priority 2 to traffic class MAPPING. When set to n, User Priority 2 is bound to Traffic Class n. Used when receiving a Priority Flow Control frame with the Timer 2 field and Class Enable Vector bit 2 set, to determine which traffic class shall be paused.
UP3MAP	11:9	0x0	RW	NA	Transmit User Priority 3 to traffic class MAPPING. When set to n, User Priority 3 is bound to Traffic Class n. Used when receiving a Priority Flow Control frame with the Timer 3 field and Class Enable Vector bit 3 set, to determine which traffic class shall be paused.
UP4MAP	14:12	0x0	RW	NA	Transmit User Priority 4 to traffic class MAPPING. When set to n, User Priority 4 is bound to Traffic Class n. Used when receiving a Priority Flow Control frame with the Timer 4 field and Class Enable Vector bit 4 set, to determine which traffic class shall be paused.
UP5MAP	17:15	0x0	RW	NA	Transmit User Priority 5 to traffic class MAPPING. When set to n, User Priority 5 is bound to Traffic Class n. Used when receiving a Priority Flow Control frame with the Timer 5 field and Class Enable Vector bit 5 set, to determine which traffic class shall be paused.
UP6MAP	20:18	0x0	RW	NA	Transmit User Priority 6 to traffic class MAPPING. When set to n, User Priority 6 is bound to Traffic Class n. Used when receiving a Priority Flow Control frame with the Timer 6 field and Class Enable Vector bit 6 set, to determine which traffic class shall be paused.
UP7MAP	23:21	0x0	RW	NA	Transmit User Priority 7 to traffic class MAPPING. When set to n, User Priority 7 is bound to Traffic Class n. Used when receiving a Priority Flow Control frame with the Timer 7 field and Class Enable Vector bit 7 set, to determine which traffic class shall be paused.
RESERVED	31:24	0x0	RSV	NA	Reserved.

8.2.2.11.5 DCB Receive Packet plane T4 Config - RTRPT4C[n] (0x00002140 + 0x4*n, n=0...7)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
CRQ	8:0	0x0	RW	NA	Credit Refill Quantum. Amount of credits to refill in 64B granularity Possible values 0x000 - 0x1FF (0 - 32,704B)
BWG	11:9	0x0	RW	NA	Bandwidth Group Index. Assignment of this PB to a bandwidth group.
MCL	23:12	0x0	RW	NA	Max Credit Limit. Max amount of credits for a configured PB in 64B granularity Possible values 0x000 - 0xFFFF (0 - 262,080B)
RESERVED	29:24	0x0	RSV	NA	Reserved.
GSP	30	0b	RW	NA	Group Strict Priority. When set to 1 enables strict priority to the appropriate PB over any traffic of other PBs within the Group.
LSP	31	0b	RW	NA	Link Strict Priority. If set to 1 enables strict priority to the appropriate PB over any traffic of other PBs.

8.2.2.11.6 Strict Low Latency Tx Queues - TXLLQ[n] (0x000082E0 + 0x4*n, n=0...3)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
STRICT_LOW_LATENCY	31:0	0x0	RW	NA	Strict Low Latency Enable. When set, defines the relevant Tx queue as strict low latency. All queues that belong to the LSP TC must be set as strict low latency queues. Bit 'm' in register 'n' corresponds to Tx queue 32 x 'n' + 'm'.

8.2.2.11.7 DCB Transmit Descriptor Plane T2 Config - RTTDT2C[n] (0x00004910 + 0x4*n, n=0...7)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
CRQ	8:0	0x0	RW	NA	Credit Refill Quantum. Amount of credits to refill the TC in 64-byte granularity Possible values 0x000 - 0x1FF (0 - 32,704 bytes)
BWG	11:9	0x0	RW	NA	Bandwidth Group Index. Assignment of this TC to a bandwidth group.
MCL	23:12	0x0	RW	NA	Max Credit Limit. Max amount of credits for a configured TC in 64-byte granularity Possible values 0x000 - 0xFFFF (0 - 262,080 bytes)
RESERVED	29:24	0x0	RSV	NA	Reserved.
GSP	30	0b	RW	NA	Group Strict Priority. When set to 1b enables strict priority to the appropriate TC over any traffic of other TCs within the group.
LSP	31	0b	RW	NA	Link Strict Priority. When set to 1b enables strict priority to the appropriate TC over any traffic of other TCs.

8.2.2.11.8 DCB Transmit Packet plane T2 Config - RTTPT2C[n] (0x0000CD20 + 0x4*n, n=0...7)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
CRQ	8:0	0x0	RW	NA	Credit Refill Quantum. Amount of credits to refill the TC in 64B granularity Possible values 0x000 - 0x1FF (0 - 32,704B)
BWG	11:9	0x0	RW	NA	Bandwidth Group. Assignment of this TC to a bandwidth group.
MCL	23:12	0x0	RW	NA	Max Credit Limit. Max amount of credits for a configured TC in 64B granularity Possible values 0x000 - 0xFFFF (0 - 262,080B)
RESERVED	29:24	0x0	RSV	NA	Reserved.
GSP	30	0b	RW	NA	Group Strict Priority. When set to 1 enables strict priority to the appropriate TC over any traffic of other TCs within the Group.
LSP	31	0b	RW	NA	Link Strict Priority. When set to 1 enables strict priority to the appropriate TC over any traffic of other TCs.

8.2.2.11.9 DCB Transmit QCN Rate-Scheduler MMW - RTTQCNRM (0x00004980)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MMW_SIZE	10:0	0x0	RW	NA	Max Memory Window SIZE for the QCN rate-scheduler, for all Tx queues. It is the maximum amount of 1KB units of payload compensation time that can be accumulated for Tx queues attached to TCn. This number shall be multiplied by the QCN Rate-Factor of the Tx queue before performing the MMW saturation check for that queue.
RESERVED	31:11	0x0	RSV	NA	Reserved.

8.2.2.11.10 DCB Transmit Descriptor Plane Queue Select - RTTDQSEL (0x00004904)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TXDQ_IDX	6:0	0x0	RW	NA	Tx Descriptor Queue Index or TX Pool of Queues Index. This register is used to set VM and BQCN transmit scheduler parameters that are configured per Tx queue or per Tx pool of queues via indirect access. It means that prior to read or write access such registers, software has to make sure this field contains the index of the Tx queue or Tx pool of queue to be accessed. When DCB is disabled, VM parameters include a pool of Tx queues. As a result, this field points to the index of the pool (and not a queue index). When DCB is enabled, and/or when programming rate limiters, this field points to a Tx queue index. The registers concerned by this index are: RTTDT1C, RTTQCNRC, RTTQCNRS
RESERVED	31:7	0x0	RSV	NA	Reserved.

8.2.2.11.11 DCB Transmit Descriptor Plane T1 Config - RTTDT1C (0x00004908)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
CRQ	13:0	X	RW	NA	Credit Refill Quantum. Amount of credits to refill the VM in 64B granularity Possible values 0x000 - 0x3FFF (0 - 1,048,512B)
RESERVED	30:14	0x0	RSV	NA	Reserved.
RESERVED	31	0b	RSV	NA	Reserved.

8.2.2.11.12 DCB Transmit QCN Rate-Scheduler Config - RTTQCNRC (0x00004984)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RF_DEC	13:0	X	RW	NA	Tx QCN rate-scheduler Rate Factor hexadecimal part, for the Tx queue indexed by TXDQ_IDX field in RTTDQSEL register Rate Factor bits that come after the hexadecimal point. Meaningful only if RS_ENA bit is set.
RF_INT	23:14	X	RW	NA	Tx QCN rate-scheduler Rate Factor Integral Part, for the Tx queue indexed by TXDQ_IDX field in RTTDQSEL register Rate Factor bits that come before the hexadecimal point. Rate Factor is defined as the ratio between the nominal link rate (i.e. 10Gbps) and the maximum rate allowed to that queue via QCN. Minimum allowed bandwidth share for a queue is 0.1% of the link rate, i.e. 10Mbps for devices operated at 10Gbps, leading to a maximum allowed Rate Factor of 1000. Meaningful only if RS_ENA bit is set.
RESERVED	30:24	0x0	RSV	NA	Reserved.
RS_ENA	31	0b	SC	NA	Tx QCN Rate-Scheduler Enable, for the Tx queue indexed by TXDQ_IDX field in RTTDQSEL register. When set, the QCN rate programmed in this register is enforced, i.e. the queue is rate controlled for QCN. At the time it is set, the current timer value is loaded into the Timestamp stored for that entry. When cleared, the QCN Rate Factor programmed in this register is meaningless, the switch for that queue is always forced to "on". The queue is not rate-controlled for QCN. Bandwidth Group Assignment of this TC to a bandwidth group. Each TC must be assigned to a different BWG number, unless the TC is a member of a BWG. No more than two TCs can share the same BWG.

8.2.2.11.13 DCB Transmit QCN Rate-Scheduler Status - RTTQCNRS (0x00004988)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MIFS	31:0	0x0	RW	NA	Tx QCN rate-scheduler current Minimum Inter-Frame Spacing, for the Tx queue indexed by TXDQ_IDX field in RTTDQSEL register. When read, it is the current algebraic value of the MIFS interval for the queue, expressed in byte units (31 LS-bits taken), relative to the QCN rate-scheduler. It is obtained by hardware subtracting the current value of the timer associated to that QCN rate-scheduler from the Timestamp stored for that queue. A strict positive value means a switch in "off" state. It is expressed in 2's complement format.

8.2.2.11.14 DCB Transmit QCN Tagging - RTTQCNTG (0x00004A90)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
CNTGI	7:0	0x0	RW	NA	CN-tag Insertion enable per TC bitmap. When bit n is cleared, hardware will not insert any CN-tag in the transmitted frame (rate-controlled or not) sent from the TCn. It is assumed that QCN is disabled for the User Priorities mapped to that TC. When bit n is set, hardware will insert the proper CN-tag in transmitted frames (rate-controlled or not) sent from the TCn, as specified in Section 7.7.2.1. It is assumed that QCN is enabled for the User Priorities mapped to that TC.
RESERVED	31:8	0x0	RSV	NA	Reserved.

8.2.2.11.15 DCB Transmit QCN Rate Reset - RTTQCNRR (0x0000498C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	0	0b	RSV	NA	Reserved.
QCN_CLEAR_ALL	1	0b	RW	NA	Clear all QCN rate-limiters. When set, the 128 RTTQCNRR.RS_ENA bits are cleared - releasing any active QCN rate-limiter. This bit shall be set by software whenever the link speed has changed. This bit is self cleared by hardware.
RESERVED	31:2				Reserved

8.2.2.12 TPH Registers

8.2.2.12.1 Rx TPH Control Register Alias - TPH_RXCTRL_ALIAS[n] (0x00002200 + 0x4*n, n=0...15)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RX_DESC_FETCH_TPH_EN	0	0b	RW	NA	Receive Descriptor Fetch TPH Enable When set, hardware enables TPH for all Rx descriptors fetch from memory. When cleared, hardware does not enable TPH for descriptor fetches. This bit is cleared as a default.
RX_DESC_WRITEBACK_TPH_EN	1	0b	RW	NA	Receive Descriptor Write-back TPH Enable When set, hardware enables TPH for all Rx descriptors written back into memory. When cleared, hardware does not enable TPH for descriptor write-backs. This bit is cleared as a default. The hint used is the hint set in the Socket ID field.
RX_HEADER_TPH_EN	2	0b	RW	NA	Receive Header TPH Enable When set, hardware enables TPH for all received header buffers. When cleared, hardware does not enable TPH for Rx headers. This bit is cleared as a default. The hint used is the hint set in the Socket ID field.
RX_PAYLOAD_TPH_EN	3	0b	RW	NA	Receive Payload TPH Enable When set, hardware enables TPH for all Ethernet payloads written into memory. When cleared, hardware does not enable TPH for Ethernet payloads. This bit is cleared as a default. The hint used is the hint set in the Socket ID field.
RESERVED	8:4	0x0	RSV	NA	Reserved.
RXDESCREADROEN	9	1b	RW	NA	Rx descriptor Read Relax Order Enable
RESERVED	10	0b	RSV	NA	
RXDESCRWBROEN	11	0b	RW	NA	Rx Descriptor Write Back Relax Order Enable. This bit must be 0b to allow correct functionality of the descriptors write-back.
RESERVED	12	0b	RSV	NA	
RXDATAWRITEROEN	13	1b	RW	NA	Rx data Write Relax Order Enable
RESERVED	14	0b	RSV	NA	
RXREPHEADERROEN	15	1b	RW	NA	Rx Split Header Relax Order Enable. This bit impacts only the Relax Order setting of the header part writes. The Relax Order setting of the data part is controlled via the RXDATAWRITEROEN bit
RESERVED	23:16	0x0	RSV	NA	
CPUID	31:24	0x00	RW	NA	Physical ID TPH capable platforms - the device driver programs a value, based on the relevant Socket ID, associated with this receive queue. Note: For TPH platforms, bits 31:27 of this field should always be set to zero.

Notes:

- Additional address(es): $0x0D00C + 0x40 \times (n-64)$, $n=64...127$
- TPH_RXCTRL[0...15] are also mapped to address $0x02200$.

8.2.2.12.2 Rx TPH Control Register - TPH_RXCTRL[n] ($0x0000D00C + 0x40 \times n$, $n=0...63$ and $0x0000100C + 0x40 \times (n-64)$, $n=64...127$)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RX_DESC_FETCH_TPH_EN	0	0b	RW	NA	Receive Descriptor Fetch TPH Enable When set, hardware enables TPH for all Rx descriptors fetch from memory. When cleared, hardware does not enable TPH for descriptor fetches. This bit is cleared as a default.
RX_DESC_WRITE_BACK_TPH_EN	1	0b	RW	NA	Receive Descriptor Write-back TPH Enable When set, hardware enables TPH for all Rx descriptors written back into memory. When cleared, hardware does not enable TPH for descriptor write-backs. This bit is cleared as a default. The hint used is the hint set in the Socket ID field.
RX_HEADER_TPH_EN	2	0b	RW	NA	Receive Header TPH Enable When set, hardware enables TPH for all received header buffers. When cleared, hardware does not enable TPH for Rx headers. This bit is cleared as a default. The hint used is the hint set in the Socket ID field.
RX_PAYLOAD_TPH_EN	3	0b	RW	NA	Receive Payload TPH Enable When set, hardware enables TPH for all Ethernet payloads written into memory. When cleared, hardware does not enable TPH for Ethernet payloads. This bit is cleared as a default. The hint used is the hint set in the Socket ID field.
RESERVED	8:4	0x0	RSV	NA	Reserved.
RXDESCREADROEN	9	1b	RW	NA	Rx descriptor Read Relax Order Enable
RESERVED	10	0b	RSV	NA	
RXDESCRWBROEN	11	0b	RW	NA	Rx Descriptor Write Back Relax Order Enable. This bit must be 0b to allow correct functionality of the descriptors write-back.
RESERVED	12	0b	RSV	NA	
RXDATAWRITEROEN	13	1b	RW	NA	Rx data Write Relax Order Enable
RESERVED	14	0b	RSV	NA	
RXREPHEADEROEN	15	1b	RW	NA	Rx Split Header Relax Order Enable. This bit impacts only the Relax Order setting of the header part writes. The Relax Order setting of the data part is controlled via the RXDATAWRITEROEN bit
RESERVED	23:16	0x0	RSV	NA	

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
CPUID	31:24	0x00	RW	NA	Physical ID TPH capable platforms - the device driver programs a value, based on the relevant Socket ID, associated with this receive queue. Note: For TPH platforms, bits 31:27 of this field should always be set to zero.

Notes:

- Additional address(es): 0x0D00C + 0x40*(n-64), n=64...127
- TPH_RXCTRL[0...15] are also mapped to address 0x02200.

8.2.2.12.3 Tx TPH Control Registers - TPH_TXCTRL[n] (0x0000600C + 0x40*n, n=0...127)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TX_DESC_FE TCH_TPH_EN	0	0b	RW	NA	Transmit Descriptor Fetch TPH Enable When set, hardware enables TPH for all Tx descriptors fetch from memory. When cleared, hardware does not enable TPH for descriptor fetches. This bit is cleared as a default.
TX_DESC_WB _TPH_EN	1	0b	RW	NA	Transmit Descriptor Write-back TPH Enable When set, hardware enables TPH for all Tx descriptors written back into memory. When cleared, hardware does not enable TPH for descriptor write-backs. This bit is cleared as a default. The hint used is the hint set in the Socket ID field.
RESERVED	2	0b	RSV	NA	Reserved.
TX_PACKET_ TPH_EN	3	0b	RW	NA	Transmit Packet TPH Enable When set, hardware enables TPH for all Ethernet payloads read from memory. When cleared, hardware does not enable TPH for Ethernet payloads. This bit is cleared as a default.
RESERVED	8:4	0x0	RSV	NA	Reserved.
TXDESCREAD ROEN	9	1b	RW	NA	Tx Descriptor Read Relax Order Enable
RESERVED	10	0b	RSV	NA	Reserved.
TXDESCRWB ROEN	11	0b	RW	NA	Tx Descriptor Write Back Relax Order Enable
RESERVED	12	0b	RSV	NA	Reserved.
TXDATAREAD ROEN	13	1b	RW	NA	Tx data read Relax Order Enable
RESERVED	23:14	0x0	RSV	NA	Reserved.
CPUID	31:24	0x0	RW	NA	Physical ID TPH capable platforms - the device driver programs a value, based on the relevant Socket ID, associated with this transmit queue. Note: For TPH platforms, bits 31:27 of this field should always be set to zero.

8.2.2.13 Timers Registers

8.2.2.13.1 TCP Timer - TCPTIMER (0x0000004C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
DURATION	7:0	0x0	RW	NA	Duration of the TCP interrupt interval, in msec.
KICKSTART	8	0b	RWS	NA	Counter kick-start Writing 1 to this bit kick-starts the counter down-count from the initial value defined in Duration field. Writing 0 has no effect.
TCPCOUNTEN	9	0b	RW	NA	TCP count enable When 1, TCP timer counting is enabled. When 0, it is disabled. Upon enabling, TCP counter shall count from its internal state. If the internal state is equal to zero, down-count does not restart until KickStart is activated. If the internal state is not 0b, down-count continues from the internal state. This enables a pause of the counting for debug purposes.
TCPCOUNTFINISH	10	0b	RWS	NA	TCP Count Finish This bit enables software to trigger a TCP timer interrupt, regardless of the internal state. Writing 1 to this bit triggers an interrupt, and resets the internal counter to its initial value. Down-count does not restart until either KickStart is activated or Loop is set. Writing 0b to this bit has no effect.
LOOP	11	0b	RW	NA	TCP Loop When 1, TCP counter shall reload duration each time it reaches zero, and shall go on down-counting from this point without kick-starting. When 0, TCP counter shall stop at zero value, and shall not re-start until KickStart is activated.
RESERVED	31:12	0x0	RSV	NA	Reserved.

8.2.2.14 Flow Director Registers

These registers are used to control the flow director functionality.

8.2.2.14.1 Flow Director Filters control register - FDIRCTRL (0x0000EE00)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PBALLOC	1:0	0x0	RW	NA	Memory allocation for the Flow Director Filters. 00 No memory allocation - Flow Director Filters are disabled 01 64K Byte (8K minus 1 signature filters or 2K minus 1 perfect match filters) 10 128K Byte (16K minus 1 signature filters or 4K minus 1 perfect match filters) 11 256K Byte (32K minus 1 signature filters or 8K minus 1 perfect match filters)
RESERVED	2	0b	RSV	NA	Reserved.
INIT_DONE	3	0b	RO	NA	Flow Director initialization completion indication (Read Only status). Indicates that hardware initialized the Flow Director table according to the PBALLOC setting. Software must not access any other Flow Director Filters registers before the INIT-Done bit is set. When Flow Director Filters are enabled (PBALLOC > 0), the software must wait for INIT-Done indication before Rx is enabled.
PERFECT_MATCH	4	0b	RW	NA	Flow Director Filters Mode of operation. When set to 1b, hardware support perfect match filters according to the PBALLOC. When cleared to 0b, hardware support Signature filters according to the PBALLOC.
REPORT_STATUS	5	0b	RW	NA	Report Flow Director Filter's status in the RSS field of the Rx descriptor for packets that matches a Flow Director filter. Enabling the Flow Director filter's status, the RXCSUM.PCSD bit should be set as well (disabling the fragment checksum). Note: The Flow Director filter Status and Error bits in the Extended Status and Error fields in the Rx descriptor are always enabled.
RESERVED	6	0b	RSV	NA	Reserved.
REPORT_STATUS_ALWAYS	7	0b	RW	NA	Report Flow Director Status in the RSS field of the Rx descriptor on any packet that can be candidates for the Flow Director filters. This bit can be set to 1 only when both the RXCSUM.PCSD bit and the Report-Status bit in this register are set.
DROP_QUEUE	14:8	0x0	RW	NA	Absolute Rx queue index used for the dropped packets. Software might set this queue to an empty one by setting the RDLEN[n] to 0x0.
DROP_NO_MATCH	15	0b	RW	NA	If set, send to the DROP_QUEUE packets that where candidate for Flow Director and did not match any filters. Candidature of packets is defined in Candidate for Flow Director per mode table in Flow Director Filters section.
FLEX_OFFSET	20:16	0x0	RW	NA	Offset within the first 64 bytes of the packet of a flexible 2 byte tuple. The offset is defined in word units counted from the first byte of the destination Ethernet MAC Address.

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FILTERMODE	23:21	0x0	RW	NA	Filter Mode: Defines the fields on which the Flow director acts: 000b: IPMODE - L3/L4 tuple. 001b: MACVLANMODE - acts on the MAC and VLAN 010b: Cloud: NVGRE/Geneve - based on TNI, inner MAC, inner VLAN. VXLAN - based on VNI, inner MAC, inner VLAN 011b - 111b: Reserved.
MAX_LENGTH	27:24	0x0	RW	NA	Max linked list length. This field defines the maximum recommended linked list associated to any Hash value. Packets that match filters that exceed MAX_LENGTH are reported with an active "Length" bit in the Extended Error field. In addition, "Drop" filters that exceed the MAX_LENGTH are posted to the Rx queue defined in the filter context rather than the DROP-Queue defined in this register. MAX_LENGTH is defined in units of 2 filters and exceeding it reflects the addition of two more filters. Note: Software should set this field to a value that indicates exceptional long buckets. Supporting 32K filters with good hash scheme key, it is expected that a value of 0xA might be a good choice.
FULL_THRESH	31:28	0x0	RW	NA	Full threshold is a recommended minimum number of flows that should remain unused (defined in units of 16 filters). When software exceeds this threshold (too low number of unused flows), hardware generates the Flow Director Full interrupt. Software should avoid additional programming following this interrupt.

8.2.2.14.2 Flow Director Filters Lookup Table HASH Key - FDIRHKEY (0x0000EE68)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
KEY	31:0	0x80000001	RW	NA	Programmable Hash lookup table Key.

8.2.2.14.3 Flow Director Filters Signature HASH Key - FDIRSKEY (0x0000EE6C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
KEY	31:0	0x80800101	RW	NA	Programmable Signature Key.

8.2.2.14.4 Flow Director Filters DIPv4 Mask - FDIRDIP4M (0x0000EE3C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
IPM	31:0	0x0	RW	NA	Mask Destination IPv4 address. Each cleared bit means that the associated bit of the destination IPv4 address is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the destination IPv4 address is ignored (masked out). The LS bit of this register matches the first byte on the wire.

8.2.2.14.5 Flow Director Filters Source IPv4 Mask - FDIRSIP4M (0x0000EE40)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
IPM	31:0	0x0	RW	NA	Mask Source IPv4 address. Each cleared bit means that the associated bit of the source IPv4 address is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the source IPv4 address is ignored (masked out). The LS bit of this register matches the first byte on the wire.

8.2.2.14.6 Flow Director Filters TCP Mask - FDIRTCPM (0x0000EE44)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
SPORTM	15:0	0x0	RW	NA	Mask TCP Source Port. Each cleared bit means that the associated bit of the TCP Source Port is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the TCP Source Port is ignored (masked out). Note: This register is swizzle as follows: bit 0 in the mask affects bit 15 of the source port as defined in FDIRPORT.Source, bit 1 in the mask affects bit 14 in FDIRPORT.Source and so on while bit 15 in the mask affects bit 0 in FDIRPORT.Source.
DPORTM	31:16	0x0	RW	NA	Mask TCP Destination Port. Each cleared bit means that the associated bit of the TCP destination Port is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the TCP destination Port is ignored (masked out). Note: This register is swizzle the same as the FDIRTCPM.SPortM.

8.2.2.14.7 Flow Director Filters UDP Mask - FDIRUDPM (0x0000EE48)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
SPORTM	15:0	0x0	RW	NA	Mask UDP Source Port. Each cleared bit means that the associated bit of the UDP Source Port is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the UDP Source Port is ignored (masked out). Note: This register is swizzle the same as the FDIRTCPM.SPortM.
DPORTM	31:16	0x0	RW	NA	Mask UDP Destination Port. Each cleared bit means that the associated bit of the UDP destination Port is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the UDP destination Port is ignored (masked out). Note: This register is swizzle the same as the FDIRTCPM.SPortM.

8.2.2.14.8 Flow Director Filters SCTP Mask - FDIRSCTPM (0x0000EE78)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
SPORTM	15:0	0x0	RW	NA	Mask SCTP Source Port. Each cleared bit means that the associated bit of the SCTP Source Port is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the SCTP Source Port is ignored (masked out). Note: This field is layout as follows: bit 0 in the mask affects bit 15 of the source port as defined in FDIRPORT.Source. bit 1 in the mask affects bit 14 in FDIRPORT.Source bit 15 in the mask affects bit 0 in FDIRPORT.Source.
DPORTM	31:16	0x0	RW	NA	Mask SCTP Destination Port. Each cleared bit means that the associated bit of the SCTP destination Port is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the SCTP destination Port is ignored (masked out). Note: This field is layout as follow: bit 0 in the mask affects bit 15 of the source port as defined in FDIRPORT.DESTINATION. bit 1 in the mask affects bit 14 in FDIRPORT.DESTINATION bit 15 in the mask affects bit 0 in FDIRPORT.DESTINATION.

8.2.2.14.9 Flow Director Filters IPv6 Mask - FDIRIP6M (0x0000EE74)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
SIPM	15:0	0x0	RW	NA	Mask Source IPv6 address. Each cleared bit means that the associated byte of the source IPv6 address is meaningful for the filtering functionality. Each bit set to 1b means that the associated byte of the source IPv6 address is ignored (masked out). The LS bit of this register matches the first byte on the wire. When working in MACVLAN mode, this field defines the mask to be used to compare the MAC address. When working in VXLAN or NVGRE mode, should be used to allow comparison of TNI/VNI and inner MAC address:.
DIPM	31:16	0x0	RW	NA	Mask Destination IPv6 address. Each cleared bit means that the associated byte of the destination IPv6 address is meaningful for the filtering functionality. Each bit set to 1b means that the associated byte of the destination IPv6 address is ignored (masked out). The whole field is meaningful only for the Hash function and the Signature based filters. The DIPv6 bit in the FDIRM register is meaningful for perfect match filters. The LS bit of this register matches the first byte on the wire.

8.2.2.14.10 Flow Director Filters Other Mask - FDIRM (0x0000EE70)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VLANID	0	0b	RW	NA	Mask VLAN ID tag. When cleared the 12 bits of the VLAN ID tag are meaningful for the filtering functionality.
VLANP	1	0b	RW	NA	Mask VLAN Priority tag. When cleared the 3 bits of the VLAN Priority are meaningful for the filtering functionality.
POOL	2	0b	RW	NA	Mask Pool. When cleared the target Pool number is meaningful for the filtering functionality.
L4P	3	0b	RW	NA	Mask L4 Protocol. When cleared the UDP/TCP/SCTP protocol type is meaningful for the filtering functionality. Note: For the Flow Director filtering aspects, SCTP is treated as if it is TCP.
FLEX	4	0b	RW	NA	Mask Flexible Tuple. When cleared the 2 bytes of the Flexible Tuple are meaningful for the filtering functionality.
DIPV6	5	0b	RW	NA	Mask Destination IPv6. When cleared the compare against the IP6AT filter is meaningful for IPv6 packets.
L3P	6	0b	RW	NA	Mask IP type comparison (In order to block all IP comparison, the DIPV6 field, the FDIRIP6M and FDIRSIP4M and FDIRDIP4M should be set also).
RESERVED	31:7	0x0	RSV	NA	Reserved.

8.2.2.14.11 Flow Director Filters Free - FDIRFREE (0x0000EE38)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FREE	15:0	0x0000	RW	NA	Number of free (non programmed) filters in the Flow Director Filters logic.
RESERVED	31:16	0x0	RSV	NA	Reserved.

8.2.2.14.12 Flow Director Filters Length - FDIRLEN (0x0000EE4C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MAXLEN	5:0	0x0	RC	NA	Longest Linked list of filters in the table. This field records the length of the longest linked list that is updated since the last time this register was read by the software. The longest bucket reported by this field includes MAXLEN + 1 filters.
RESERVED	7:6	0x0	RSV	NA	Reserved.
BUCKET_LEN GTH	13:8	0x0	RC	NA	The length of the Linked list indicated by a query command. This field is valid following a query command completion.
RESERVED	15:14	0x0	RSV	NA	Reserved.
MAXHASH	30:16	0x0	RC	NA	The Lookup HASH value of the added filter that updated the value of the MAXLEN field in this register.
RESERVED	31	0b	RSV	NA	Reserved.

8.2.2.14.13 Flow Director Filters Usage Statistics - FDIRUSTAT (0x0000EE50)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
ADD	15:0	0x0	RC	NA	Number of added filters. This field counts the number of added filters to the Flow Director Filters logic. The counter is stuck at 0xFFFF and cleared on read.
REMOVE	31:16	0x0	RC	NA	Number of removed filters. This field counts the number of removed filters to the Flow Director Filters logic. The counter is stuck at 0xFFFF and cleared on read.

8.2.2.14.14 Flow Director Filters Failed Usage Statistics - FDIRFSTAT (0x0000EE54)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FADD	7:0	0x0	RC	NA	Number of failed added filters due to no space in the filter table. The counter is stuck at 0xFF and cleared on read.
FREMOVE	15:8	0x0	RC	NA	Number of failed removed filters. The counter is stuck at 0xFF and cleared on read.
RESERVED	31:16	0x0	RSV	NA	Reserved.

8.2.2.14.15 Flow Director Filters Match Statistics - FDIRMATCH (0x0000EE58)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PCNT	31:0	0x0	RC	NA	Number of packets that matched any Flow Director filter. The counter is stacked at 0xFF...F and cleared on read. Note: This counter might include packets that match the L2 filters or 5 tuple filters or Syn filters even if they are enabled for queue assignment.

8.2.2.14.16 Flow Director Filters Miss Match Statistics - FDIRMISS (0x0000EE5C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PCNT	31:0	0x0	RC	NA	Number of packets that missed matched any Flow Director filter. The counter is stacked at 0xFF...F and cleared on read.

Note: Flow Programming Registers

8.2.2.14.17 Flow Director Filters Source IPv6 - FDIRSIPV6[n] (0x0000EE0C + 0x4*n, n=0...2)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
IP6SA	31:0	0x0	RW	NA	3 LS DWords of the Source IPv6 address. The FDIRIPSA contains the MS DWord of the IP6 address. The LS byte of FDIRIPSA is first on the wire and the MS byte of FDIRSIPV6[2] is last on the wire In MACVLANMODE FDIRSIPV6_0[31:0], FDIRSIPV6_1[15:0] hold the destination MAC address value FDIRSIPV6_1[31:16], FDIRSIPV6_2[31:0] are reserved and should be set to 0x0

8.2.2.14.18 Flow Director Filters IP SA - FDIRIPSA (0x0000EE18)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
IP4SA	31:0	0x0	RW	NA	Source IPv4 address or MS DWord of the Source IPv6 address. While the field is defined in Big Endian (LS byte is first on the wire). In MACVLANMODE this field is reserved and should be set to 0x0.

8.2.2.14.19 Flow Director Filters IP DA - FDIRIPDA (0x0000EE1C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
IP4DA	31:0	0x0	RW	NA	Destination IPv4 address. While the field is defined in Big Endian (LS byte is first on the wire). In MACVLANMODE this field is reserved and should be set to 0x0.

8.2.2.14.20 Flow Director Filters Port - FDIRPORT (0x0000EE20)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
SOURCE	15:0	0x0	RW	NA	Source Port number while the field is defined in Little Endian (MS byte is first on the wire). In MACVLANMODE and cloud modes (NVGRE/VXLAN) this field is reserved and should be set to 0x0.
DESTINATION	31:16	0x0	RW	NA	Destination Port number while the field is defined in Little Endian (MS byte is first on the wire). In MACVLANMODE and cloud modes (NVGRE/VXLAN/Geneve) this field is reserved and should be set to 0x0.

8.2.2.14.21 Flow Director Filters VLAN and FLEX bytes - FDIRVLAN (0x0000EE24)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VLAN	15:0	0x0	RW	NA	VLAN Tag: This field is defined in Little Endian (MS byte is first on the wire). The CFI bit must be set to Zero and it is not checked by hardware.
FLEX	31:16	0x0	RW	NA	Flexible tuple data as defined by the Flex-Offset field in the FDIRCTRL register while the field is defined in Big Endian (LS byte is first on the wire).

8.2.2.14.22 Flow Director Filters Hash Signature - FDIRHASH (0x0000EE28)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
HASH	14:0	0x0	RW	NA	Bucket hash value that identifies a Filter's linked list
BUCKET_VALID	15	0b	RW	NA	The Valid bit is set by hardware whenever there is at least one filter assigned to this Hash.
SIGNATURE_SW_INDEX	30:16	0x0	RW	NA	Flow Director Filter Signature for Signature filters and SW-index for perfect match filters. In perfect match mode, the two MSbits should be zero.
RESERVED	31	0b	RSV	NA	Reserved.

8.2.2.14.23 Flow Director Filters Command register - FDIRCMD (0x0000EE2C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
CMD	1:0	0x0	RW	NA	Flow Director Filter Programming Command: 00b No Action 01b Add Flow 10b Remove Flow 11b Query Command. Following a command completion hardware clears the CMD field. In a query command, all other parameters are valid when the CMD field is zero.
FILTER_VALID	2	0b	RW	NA	Valid Filter is found by the query command. This bit is set by the device following a Query Command completion.
FILTER_UPDATE	3	0b	RW	NA	Filter Update Command. This bit is relevant only for "Add Flow" command and must be set to zero in any other commands. When cleared, the filter parameters do not override existing ones if exist while setting only the collision bit. When set to '1' the new filter parameters override existing ones if exist keeping the collision bit as is.
IPV6DMATCH	4	0b	RW	NA	IP Destination match to IP6AT filter. This bit is meaningful only for perfect match IPv6 filters. Otherwise it should be cleared by software at programming time. When set to 1b the Destination IPv6 address should match the IP6AT. When cleared, the Destination IPv6 address should not match the IP6AT. This field might never match local VM to VM traffic.
L4TYPE	6:5	0x0	RW	NA	L4 Packet Type. Defines the packet as one of the following L4 types: 00 Reserved. 01 UDP 10 TCP 11 SCTP
IPV6	7	0b	RW	NA	IPv6 packet type when set to 1b and IPv4 packet type at 0b. Relevant only if FDIRM.L3P is not set.
CLEARHT	8	0b	RW	NA	Clear the Flow Director Head and Tail Registers. This bit is set only as part of Flow Director init. During nominal Operation it must be kept at 0b.

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	10:9				Reserved
LAST	11	0b	RW	NA	Last filter indication in the linked list. At Flow programming the software should set the Last bit to 1b. Hardware might modify this bit when adding or removing flows from the same linked list.
COLLISION	12	0b	RW	NA	Collision indication. This field is set to one when software programs the same filter multiple times. In Signature based filtering it is set when software programs a filter with the same Hash and Signature multiple times. It should be cleared by software when it adds a Flow. It might be set by hardware when 2 flows collide with the same Hash and signature. During reception, this bit is reported on the Rx descriptor of packets that match the filter. See bit 7 for description of the Query-Type.
RESERVED	14:13	0x0	RSV	NA	Reserved.
QUEUE_EN	15	0b	RW	NA	Enable routing matched packet to the queue defined by the Rx-Queue. Note: Packets redirection to the FDIRCTRL.Drop-Queue is not gated by the Queue-EN bit.
RX_QUEUE	22:16	0x0	RW	NA	Rx queue index. This field defines the absolute Rx queue index in all modes of operation (regardless of DCB and VT enablement).
TUNNEL_FILTER	23	0b	RSV	NA	If set, the filter will be matched by NVGRE or VXLAN packets only. In this case, the L3 and L4 parameters programmed should be of the tunneled (inner) header.
POOL	29:24	0x0	RW	NA	Pool number is meaningful when VT mode is enabled. When both VT is not enabled, this field must be set by software to 0.
RESERVED	30	0b	RSV	NA	Reserved.
FD_EXCEPTION	31	0b	RW	NA	If set after a command was given, indicates a flow director exception occurred. This bit is a reflection of the EICR.FLOW_DIRECTOR bit. Should be set to zero when a new command is given

8.2.2.15 MAC Registers

8.2.2.15.1 MAC SGMII Busy Register - MAC_SGMII_BUSY (0x00004298)

Reset source: MAC Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
HLREG0_BUSY	0	0b	RO	NA	
RESERVED0	1	0b	RO	NA	
PAP_BUSY	2	0b	RO	NA	
RESERVED	9:3	0x0	RO	NA	
MAXFRS_BUSY	10	0b	RO	NA	
TREG_BUSY	11	0b	RO	NA	
MFLCN_BUSY	12	0b	RO	NA	
RESERVED1	13	0b	RO	NA	
MPVC	14	0b	RO	NA	
PFCTOP_BUSY	15	0b	RO	NA	
FCAMACL_BUSY	16	0b	RO	NA	
FCAMACH_BUSY	17	0b	RO	NA	
MACC_BUSY	18	0b	RO	NA	
RESERVED3	31:19	0x0	RO	NA	

8.2.2.15.2 Highlander Control 0 Register - HLREG0 (0x00004240)

Reset source: MAC Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	1:0	00b	RSV	NA	Reserved
JUMBOEN	2	0b	RW	NA	JUMBO FRAME ENABLE: Allows frames up to the size specified in Reg MAXFRS (31:16) 1 - Enable Jumbo Frames 0 - Disable Jumbo Frames (Default)
RESERVED	9:3	0x7F	RSV	NA	Reserved. must be set to 1111111b.
TXPADEN	10	1b	RW	NA	Tx PAD FRAME ENABLE: Pad short Tx frames to 64 bytes if requested by user 1 - Pad Frames (Default) 0 - Transmit Short Frames With No Padding
RESERVED	11	1b	RSV	NA	Reserved. must be set to 1.
RESERVED	12				Reserved
RESERVED	13	1b	RSV	NA	Reserved. must be set to 1.
RESERVED	14				Reserved
LPBK	15	0b	RW	NA	LOOPBACK: Turn On Loopback Where Transmit Data Is Sent Back Through Receiver 1 - Loopback Enabled 0 - Loopback Disabled (Default)

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MDCSPD	16	1b	RW	NA	Controls the MDC frequency MDC frequency is dependent on the active link speed. Port Speed 10G Other MDCSPD=0 2.4MHz 240KHz MDCSPD=1 Reserved 2.4MHz Note: When ever link is down SW should configure this field to 0. SW should configure this field to 1 only when link is up and speed is not 10G.
CONTMDC	17	0b	RW	NA	CONTINUOUS MDC: Turn Off MDC Between MDIO Packets 1 - Continuous MDC 0 - MDC Off Between Packets (default)
RESERVED	19:18	00b	RSV	NA	Reserved
PREPEND	23:20	0x0	RW	NA	PREPEND VALUE: Number Of 32 Bit Words Starting after the Preamble and SFD, to Exclude From the CRC Generator And Checker (Default - 0 Hex)
RESERVED	26:24	0x0	RSV	NA	Reserved.
RXLNGTHERR EN	27	1b	RW	NA	Rx Length Error Reporting: 1 - Enable reporting of rx_length_err events if length field < 0x0600. 0 - Disable reporting of all rx_length_err events.
RXPADSTRIP EN	28	0b	RW	NA	Rx Padding Strip Enable: 1 - Strip Padding from Rx packets with length field < 64 (debug only). 0 - Do Not Strip Padding from Rx packets with length field < 64 (Default). Note: This functionality should be used as debug mode only. If Rx Pad Stripping is enabled, then the Rx CRC Stripping needs to be enabled as well.
RESERVED	31:29	0x0	RSV	NA	Reserved.

8.2.2.15.3 Highlander Status 1 Register - HLREG1 (0x00004244)

Reset source: MAC Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	3:0	0x1	RSV	NA	Reserved.
RESERVED	4	0b	RSV	NA	Reserved.
RXERRSYM	5	0b	RO	NA	Rx ERROR SYMBOL: Error Symbol During Rx Packet (Latch High, Clear On Read) 1 - Error Symbol Received 0 - No Error Symbol (Default)
RXILLSYM	6	0b	RO	NA	Rx ILLEGAL SYMBOL: Illegal Symbol During Rx Packet (Latch High, Clear On Read) 1 - Illegal Symbol Received 0 - No Illegal Symbol Received (Default)
RXIDLERR	7	0b	RO	NA	Rx IDLE ERROR: Non Idle Symbol During Idle Period (Latch High, Clear On Read) 1 - Idle Error Received 0 - No Idle Errors Received (Default)

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RXLCLFLT	8	0b	RO	NA	Rx LOCAL FAULT: Fault reported from PMD, PMA, or PCS (Latch High, Clear On Read) 1 - Local Fault Is Or Was Active 0 - No Local Fault (Default)
RXRMTFLT	9	0b	RO	NA	Rx REMOTE FAULT: Link Partner Reported Remote Fault (Latch High, Clear On Read) 1 - Remote Fault Is Or Was Active 0 - No Remote Fault (Default)
RESERVED	31:10	0x0	RSV	NA	Reserved.

8.2.2.15.4 Pause and Pace Register - PAP (0x00004248)

Reset source: MAC Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	15:0				Reserved
PACE	19:16	0x0	RW	NA	0000 - 10 Gbps (LAN) 0001 - 1 Gbps 0010 - 2 Gbps 0011 - 3 Gbps 0100 - 4 Gbps 0101 - 5 Gbps 0110 - 6 Gbps 0111 - 7 Gbps 1000 - 8 Gbps 1001 - 9 Gbps 1111 - 9.294196 Gbps (WAN)
RESERVED	31:20	0x0	RSV	NA	Reserved.

8.2.2.15.5 MDI Single Command and Address - MSCA (0x0000425C)

Reset source: MAC Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MDIADD	15:0	0x0000	RW	NA	MDI ADDRESS: Address Used For MDI Accesses
DEVADD	20:16	0x0	RW	NA	When STCODE = 00b: These are the bits representing the device type When STCODE = 01b: These bits represent the register address
PORTADD	25:21	0x0	RW	NA	PORT ADDRESS: The Address Of The PHY Port. This field must be set to the port number as reflected in STATUWS.LAN_ID field.
OPCODE	27:26	0x0	RW	NA	OP CODE: Two Bits Identifying Operation To Be Performed 00 - Address Cycle 01 - Write Operation 10 - Read, Increment Address 11 - Read Operation
STCODE	29:28	0x0	RW	NA	ST Code: Two Bits Identifying Start Of Frame And Old Or New Protocol (Default = 01). 00b = New protocol. 01b = Old protocol. 1Xb = Illegal.

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MDICMD	30	0b	RW	NA	MDI COMMAND: Used to trigger the MDI Operation, Cleared When Done 1 - Perform Operation, Operation In Progress 0 - MDI Ready, Operation Complete
RESERVED	31	0b	RSV	NA	Reserved.

8.2.2.15.6 MDI Single Read and Write Data - MSRWD (0x00004260)

Reset source: MAC Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MDIWRDATA	15:0	0x0	RW	NA	MDI WRITE DATA: Write Data.
MDIRDDATA	31:16	0x0	RW	NA	MDI READ DATA: Read Data (RO).

8.2.2.15.7 Max Frame Size - MAXFRS (0x00004268)

Reset source: MAC Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	15:0	0x0	RSV	NA	Reserved.
MFS	31:16	0x5EE	RW	NA	This field defines the Maximum Frame Size in bytes units from Ethernet MAC Addresses up to inclusive the CRC. Frames received that are larger than this value are dropped. This field is meaningful when Jumbo are enabled (HLREG0.JUMBOEN = 1). When Jumbo frames are not enabled the device uses a hard-wired value of 1518 for this field. The MFS does not include the 4 bytes of the VLAN header. Packets with VLAN header might be as large as MFS + 4. When Double VLAN is enabled the device adds 8 to the MFS for any packet. When E-Tags are enabled, the device adds 12 to the MFS for any packet. This value has no effect on transmit frames; it is the responsibility of software to limit the size of transmit frames. Note: Packets to/from MC are limited to 2KB even when Jumbo are enabled.

8.2.2.15.8 Link Status Register - LINKS (0x000042A4)

Reset source: MAC Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FIFO_UNDER RUN	0	0b	RO	NA	Indicates underrun condition in MAC elastic FIFO
FIFO_OVERR UN	1	0b	RO	NA	Indicates overrun condition in MAC elastic FIFO
RF_STATE	2	0b	RO	NA	MAC is in Remote Fault state.
LF_STATE	3	0b	RO	NA	MAC is in Local Fault state
RESERVED	26:4	0x0	RSV	NA	Reserved.
NON_STAND ARD_SPEED	27	0b	RO	NA	If set, the LINK_SPEED field reflects non standard speeds (2.5/5G)
LINK_SPEED	29:28	0x0	RO	NA	MAC link speed status If NON_STANDARD_SPEED = 0: 00b - 10 Mbps. 01b - 100 Mbps 10b - 1 Gbps 11b - 10 Gbps If NON_STANDARD_SPEED = 1: 00b - Reserved. 01b - 5 Gbps 10b - RSV 11b - 2.5 Gbps
LINK_UP	30	0b	RO	NA	Link up 1 - Link is up. 0 - Link is down.
RESERVED	31	0b	RSV	NA	Reserved.

8.2.2.15.9 MAC Manageability Control Register - MMNGC (0x000042D0)

Reset source: MAC Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MNG_VETO	0	0b	RO	NA	MNG_VETO (default 0) access read/write by the management controller, read only to the host. 0 = No specific constraints on link from management controller. 1 = Hold off any low-power link mode changes. This is done to avoid link loss and interrupting management traffic/activity.
RESERVED	31:1	0x0	RSV	NA	Reserved.

8.2.2.15.10 MAC Control register - MACC (0x00004330)

Reset source: MAC Reset, NVM Load: Type1/2 in MAC Reset Register Autoload module, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FLU	0	0b	RW	NA	Force Link Up. 0 - Normal mode 1 - MAC is forced to the link up state regardless to the PHY link status
MAC_RX2TX_LPBK_EN	1	0b	RW	NA	Enable MAC Rx to Tx loopback. 0 - No loopback, normal mode. 1 - Loopback enabled. Transmit path is driven from the receive path, at the MAC internal XGMII interface.
SWIZZLE_TX_DATA	2	0b	RW	NA	Swizzle the bytes in all 4 MAC internal Tx XGMII lanes. 0 - Swizzle disabled, normal mode. 1 - Swizzle enabled.
SWAP_TX_DATA	3	0b	RW	NA	Swap the MAC internal Tx XGMII lanes. 0 - Swap disabled, normal mode. 1 - Swap enabled.
SWAP_TX_CONTROL	4	0b	RW	NA	Swap the MAC internal Tx XGMII controls. 0 - Swap disabled, normal mode. 1 - Swap enabled.
SWIZZLE_RX_DATA	5	0b	RW	NA	Swizzle the bytes in all 4 MAC internal Rx XGMII lanes. 0 - Swizzle disabled, normal mode. 1 - Swizzle enabled.
SWAP_RX_DATA	6	0b	RW	NA	Swap the MAC internal Rx XGMII lanes. 0 - Swap disabled, normal mode. 1 - Swap enabled.
SWAP_RX_CONTROL	7	0b	RW	NA	Swap the MAC internal Rx XGMII controls. 0 - Swap disabled, normal mode. 1 - Swap enabled.
FIFOHT	11:8	0xD	RW	NA	FIFO High Threshold. Determines the high threshold of the MAC elastic FIFO.
FIFOLT	15:12	0x3	RW	NA	FIFO Low Threshold. Determines the low threshold of the MAC elastic FIFO.
RESERVED	31:16	0x0	RSV	NA	Reserved.

8.2.2.16 Statistic Registers

Notes:

- All Statistics registers are cleared on read. In addition, they stick at 0xFF...F when the maximum value is reached.
- For the receive statistics, it should be noted that a packet is indicated as received if it passes the device filters and is placed into the packet buffer memory. A packet does not have to be DMA'd to host memory in order to be counted as received.
- Due to divergent paths between interrupt generation and logging of relevant statistics counts, it might be possible to generate an interrupt to the system for a noteworthy event prior to the associated statistics count actually being incremented. This is extremely unlikely due to expected delays associated with the system interrupt collection and ISR delay, but might be an explanation for interrupt statistics values that do not quite make sense. Hardware guarantees that any event noteworthy of inclusion in a statistics count is reflected in the appropriate count within 1's; a small time-delay prior to reading the statistics might be required to avoid a potential mismatch between and interrupt and its cause.
- If TSO is enabled, statistics are collected after segmentation.
- All byte (octet) counters composed of two registers can be fetched by two consecutive 32-bit accesses while reading the low 32-bit register first or a single 64-bit access.
- All receive statistic counters in the filter unit (listed below) might count packets that might be dropped by the packet buffer or receive DMA. Same comment is valid for the byte counters associated with these packet counters: PRC64, PRC127, PRC255, PRC511, PRC1023, PRC1522, BPRC, MPRC, GPRC, RXNFGPC, RUC, ROC.

Statistics Hierarchy:

The following diagrams describe the relations between the packet flow and the different statistic counters.

- [Figure 7-13, "Receive Flow Statistics" on page 1078](#)
- [Figure 7-21, "Transmit Flow Statistics" on page 1112](#)

8.2.2.16.1 Error Packets VLAN Mismatch Counter - ERR_PKT_VLAN_MIS_CNT (0x00004044)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
ERR_PKT_VLAN_MIS_CNT	31:0	0x0	RC	NA	Counts the number of packets dropped due to VLAN filtering mismatch.

8.2.2.16.2 Error Packet Address Mismatch Counter - ERR_PKT_ADD_MIS_CNT (0x00004048)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
ERR_PKT_ADD_MIS_CNT	31:0	0x0	RC	NA	Counts the number of packets dropped due to address filtering mismatch.

8.2.2.16.3 LINK Down Counter - LINK_DN_CNT (0x0000403C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
LINK_DN_CNT	31:0	0x0	RC	NA	Counts the number of link down events in the MAC.

8.2.2.16.4 Bad SFD Count - MBSDC (0x00004018)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MBSDC	31:0	0x0	RW	NA	Counts the number of packets received with bad SFD. Does not count short packets not received

8.2.2.16.5 CRC Error Count - CRCERRS (0x00004000)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
CEC	31:0	0x0	RC	NA	CRC error count. Counts the number of receive packets with CRC errors. In order for a packet to be counted in this register, it must be 64 bytes or greater (from <Destination Address> through <CRC>, inclusively) in length. This registers counts all packets received, regardless of L2 filtering and receive enable. This register doesn't counts packets with bad sfd, error control byte or illegal code byte.

8.2.2.16.6 Illegal Byte Error Count - ILLERRC (0x00004004)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
IBEC	31:0	0x0	RC	NA	Illegal Byte error count. Counts the number of receive packets with illegal bytes errors (i.e. there is an illegal symbol in the packet). This registers counts all packets received, regardless of L2 filtering and receive enablement.

8.2.2.16.7 Error Byte Packet Count - ERBPC (0x00004008)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
EBC	31:0	0x0	RC	NA	Error Byte Packet Counter. Counts the number of received packets with Error bytes (i.e. there is an Error symbol in the packet). This registers counts all packets received, regardless of L2 filtering and receive enablement.

8.2.2.16.8 MAC Local Fault Count - MLFC (0x00004034)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MLFC	31:0	0x0	RC	NA	Number of faults in the local MAC. This register is valid only when the link speed is 10Gbps.

8.2.2.16.9 MAC Remote Fault Count - MRFC (0x00004038)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MRFC	31:0	0x0	RC	NA	Number of faults in the remote MAC. This register is valid only when the link speed is 10Gbps.

8.2.2.16.10 Receive Length Error Count - RLEC (0x00004040)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RLEC	31:0	0x0	RC	NA	Number of packets with receive length errors. A length error occurs if an incoming packet length field in the MAC header doesn't match the packet length. To enable the receive length error count HLRG0.RXLNGTHERREN bit needs to be set to 1. This registers counts all packets received, regardless of L2 filtering and receive enablement.

8.2.2.16.11 Switch Security Violation Packet Count - SSVPC (0x00008780)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
SSVPC	31:0	0x0	RC	NA	Switch Security Violation Packet Count. This register counts all Tx packets dropped, e.g. due to switch security violations such as SA or VLAN anti spoof filtering or packet that has (inner) VLAN that contradicts with PFVMVIR register definitions. Valid only in VMDq or IOV mode. This counter includes also traffic sent to the BMC and blocked due to security violations. or packets dropped due to malicious events detection.

8.2.2.16.12 Link XON Transmitted Count - LXONTXC (0x00003F60)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
XONTXC	15:0	0x0	RC	NA	Number of XON packets Transmitted. Sticks to 0xFFFF. XONTXC is incremented by one for each Link XON packet when MFLCN.RFCE is set and for each Priority XON packet when corresponding MFLCN.RPFCE bit is set.
RESERVED	31:16	0x0	RSV	NA	Reserved.

8.2.2.16.13 Link XON Received Count - LXONRXCNT (0x000041A4)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
XONRXC	15:0	0x0	RC	NA	Number of XON packets Received. Sticks to 0xFFFF. XON packets can use the global address, or the station address. This register counts any XON packet whether it is a legacy XON or a priority XON. Each XON packet is counted once even if it designated to a few priorities. If a priority FC packet contains both XOFF and XON, only the LXOFFRXCNT counter is incremented.
RESERVED	31:16	0x0	RSV	NA	Reserved.

8.2.2.16.14 Link XOFF Transmitted Count - LXOFFTXC (0x00003F68)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
XOFFTXC	15:0	0x0	RC	NA	Number of XOFF packets Transmitted. Sticks to 0xFFFF. XOFFTXC is incremented by one for each Link XOFF packet when MFLCN.RFCE is set and for each Priority XOFF packet when corresponding MFLCN.RPFCE bit is set.
RESERVED	31:16	0x0	RSV	NA	Reserved.

8.2.2.16.15 Link XOFF Received Count - LXOFFRXCNT (0x000041A8)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
XOFFRXC	15:0	0x0	RC	NA	Number of XOFF packets Received. Sticks to 0xFFFF. XOFF packets can use the global address, or the station address. This register counts any XOFF packet whether it is a legacy XOFF or a priority XOFF. Each XOFF packet is counted once even if it designated to a few priorities. If a priority FC packet contains both XOFF and XON, only this counter is incremented.
RESERVED	31:16	0x0	RSV	NA	Reserved.

8.2.2.16.16 Priority XON Transmitted Count - PXONTXC[n] (0x00003F00 + 0x4*n, n=0...7)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
XONTXC	15:0	0x0	RC	NA	Number of XON packets transmitted. per TC. Sticks to 0xFFFF.
RESERVED	31:16	0x0	RSV	NA	Reserved.

8.2.2.16.17 Priority XON Received Count - PXONRXCNT[n] (0x00004140 + 0x4*n, n=0...7)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
XONRXC	15:0	0x0	RC	NA	Number of XON packets received per UP. Sticks to 0xFFFF.
RESERVED	31:16	0x0	RSV	NA	Reserved.

8.2.2.16.18 Priority XOFF Transmitted Count - PXOFFTXCNT[n] (0x00003F20 + 0x4*n, n=0...7)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
XOFFTXC	15:0	0x0	RC	NA	Number of XOFF packets transmitted per TC. Sticks to 0xFFFF.
RESERVED	31:16	0x0	RSV	NA	Reserved.

8.2.2.16.19 Priority XOFF Received Count - PXOFFRXCNT[n] (0x00004160 + 0x4*n, n=0...7)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
XOFFRXC	15:0	0x0	RC	NA	Number of XOFF packets received per UP. Sticks to 0xFFFF.
RESERVED	31:16	0x0	RSV	NA	Reserved.

8.2.2.16.20 Priority XON to XOFF Count - PXON2OFFCNT[n] (0x00003240 + 0x4*n, n=0...7)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
XON2OFF	15:0	0x0	RC	NA	Number of times the pause state has transitioned from XON to XOFF per TC. Sticks to 0xFFFF. The register attached to User Priority 0 reports also Link Flow Control statistics.
RESERVED	31:16	0x0	RSV	NA	Reserved.

8.2.2.16.21 Packets Received [64 Bytes] Count - PRC64 (0x0000405C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PRC64	31:0	0x0	RW	NA	Number of good packets received that are 64 bytes in length (from <Destination Address> through <CRC>, inclusively). This registers counts packets that pass L2 filtering regardless on receive enablement and does not include received flow control packets.

8.2.2.16.22 Packets Received [65-127 Bytes] Count - PRC127 (0x00004060)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PRC127	31:0	0x0	RW	NA	Number of packets received that are 65-127 bytes in length (from <Destination Address> through <CRC>, inclusively). This registers counts packets that pass L2 filtering regardless on receive enablement and does not include received flow control packets.

8.2.2.16.23 Packets Received [128-255 Bytes] Count - PRC255 (0x00004064)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PRC255	31:0	0x0	RW	NA	Number of packets received that are 128-255 bytes in length (from <Destination Address> through <CRC>, inclusively). This registers counts packets that pass L2 filtering regardless on receive enablement and does not include received flow control packets.

8.2.2.16.24 Packets Received [256-511 Bytes] Count - PRC511 (0x00004068)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PRC511	31:0	0x0	RW	NA	Number of packets received that are 256-511 bytes in length (from <Destination Address> through <CRC>, inclusively). This registers counts packets that pass L2 filtering regardless on receive enablement and does not include received flow control packets.

8.2.2.16.25 Packets Received [512-1023 Bytes] Count - PRC1023 (0x0000406C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PRC1023	31:0	0x0	RW	NA	Number of packets received that are 512-1023 bytes in length (from <Destination Address> through <CRC>, inclusively). This registers counts packets that pass L2 filtering regardless on receive enablement and does not include received flow control packets.

8.2.2.16.26 Packets Received [1024 to Max Bytes] Count - PRC1522 (0x00004070)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PRC1522	31:0	0x0	RW	NA	Number of packets received that are 1024-Max bytes in length (from <Destination Address> through <CRC>, inclusively). This registers counts packets that pass L2 filtering regardless on receive enablement and does not include received flow control packets. The maximum is dependent on the current receiver configuration and the type of packet being received. If a packet is counted in Receive Oversized Count, it is not counted in this register (see Section 8.2.3.24.52). Due to changes in the standard for maximum frame size for VLAN tagged frames in 802.3, this device accepts packets which have a maximum length of 1522 bytes. The RMON statistics associated with this range has been extended to count 1522 byte long packets.

8.2.2.16.27 Broadcast Packets Received Count - BPRC (0x00004078)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
BPRC	31:0	0x0	RC	NA	Number of good (non-erred) broadcast packets received. This register does not count received broadcast packets when the broadcast address filter is disabled. The counter counts packets regardless on receive enablement.

8.2.2.16.28 Multicast Packets Received Count - MPRC (0x0000407C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MPRC	31:0	0x0	RC	NA	Number of good (non-erred) multicast packets received that pass L2 filtering (excluding Broadcast packets). This register does not count received flow control packets. This registers counts packets regardless on receive enablement.

8.2.2.16.29 Good Packets Received Count - GPRC (0x00004074)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GPRC	31:0	0x0	RC	NA	Number of good (non-erred) Rx packets (from the network) that pass L2 filtering and has legal length as defined by the LongPacketEnable. This registers counts packets regardless of receive enable, this register doesn't count flow control packets.

8.2.2.16.30 Good Octets Received Count Low - GORCL (0x00004088)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
CNT_L	31:0	0x0	RC	NA	Lower 32 bits of the "Good Octets Received" counter. The GORCL and GORCH registers make up a logical 36-bit octet counter of the packets counted by the GPRC. This register includes bytes received in a packet from the <Destination Address> field through the <CRC> field, inclusively.

8.2.2.16.31 Good Octets Received Count High - GORCH (0x0000408C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
CNT_H	3:0	0x0	RC	NA	Higher 4 bits of the "Good Octets Received" counter.
RESERVED	31:4	0x0	RSV	NA	Reserved.

8.2.2.16.32 Good Rx Non-Filtered Packet Counter - RXNFGPC (0x000041B0)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GPC	31:0	0x0	RC	NA	Number of good (non-erred with legal length) Rx packets (from the network) regardless of packet filtering and receive enablement.

8.2.2.16.33 Good Rx Non-Filter Byte Counter Low - RXNFBCL (0x000041B4)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
BCL	31:0	0x0	RC	NA	Low 32 bits of the 36 bit Byte Counter of good (non-erred) Rx packets that match the RXNFGPC. The counter counts all bytes from <Destination Address> field through the <CRC> field, inclusively.

8.2.2.16.34 Good Rx Non-Filter Byte Counter High - RXNFG BCH (0x000041B8)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
BCH	3:0	0x0	RC	NA	High 4 bits of the 36 bit Byte Counter associated to RXFGBCL.
RESERVED	31:4	0x0	RSV	NA	Reserved.

8.2.2.16.35 DMA Good Rx Packet Counter - RXDGPC (0x00002F50)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GPC	31:0	0x0	RC	NA	Number of good (non-erred) Rx packets from the Network posted to the host memory. In case of packet replication (or mirrored) the counter counts each packet only once. The counter might count packets directed to ALL Rx queues or specific Rx queues as defined by the RXDSTATCTRL register.

8.2.2.16.36 DMA Good Rx Byte Counter Low - RXDGBCL (0x00002F54)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GBCL	31:0	0x0	RC	NA	Low 32 bits of the 36 bit Byte Counter of good (non-erred) Rx packets that match the RXDGPC. The counter counts all bytes posted to the host before VLAN strip.

8.2.2.16.37 DMA Good Rx Byte Counter High - RXDGBCH (0x00002F58)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GBCH	3:0	0x0	RC	NA	High 4 bits of the 36 bit Byte Counter associated to RXDGBCL.
RESERVED	31:4	0x0	RSV	NA	Reserved.

8.2.2.16.38 DMA Duplicated Good Rx Packet Counter - RXDDPC (0x00002F5C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GPC	31:0	0x0	RC	NA	Number of replicated or mirrored packets that meet the RXDGPC conditions. The sum of RXDDPC and RXDGPC is the total good (non-erred) Rx packets from the Network that are posted to the host (see Note). Note: The counter might count packets directed to ALL Rx queues or specific Rx queues as defined by the RXDSTATCTRL register.

8.2.2.16.39 DMA Duplicated Good Rx Byte Counter Low - RXDDBCL (0x00002F60)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GBCL	31:0	0x0	RC	NA	Low 32 bits of the 36 bit Byte Counter of good (non-erred) Rx packets that match the RXDDPC. The counter counts all bytes posted to the host before VLAN strip.

8.2.2.16.40 DMA Duplicated Good Rx Byte Counter High - RXDDBCH (0x00002F64)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GBCH	3:0	0x0	RC	NA	High 4 bits of the 36 bit Byte Counter associated to RXDDBCL.
RESERVED	31:4	0x0	RSV	NA	Reserved.

8.2.2.16.41 DMA Good Rx LPBK Packet Counter - RXLPBKPC (0x00002F68)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GPC	31:0	0x0	RC	NA	Number of good (non-erred) Rx packets from a local VM posted to the host memory. In case of packet replication (or mirrored) the counter counts each packet only once. The counter might count packets directed to ALL Rx queues or specific Rx queues as defined by the RXDSTATCTRL register.

8.2.2.16.42 DMA Good Rx LPBK Byte Counter Low - RXLPBKBC (0x00002F6C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GBCL	31:0	0x0	RC	NA	Low 32 bits of the 36 bit Byte Counter of good (non-erred) Rx packets that match the RXLPBKPC. The counter counts all bytes posted to the host before VLAN strip.

8.2.2.16.43 DMA Good Rx LPBK Byte Counter High - RXLPBKBCH (0x00002F70)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GBCH	3:0	0x0	RC	NA	High 4 bits of the 36 bit Byte Counter associated to RXLPBKBCL.
RESERVED	31:4	0x0	RSV	NA	Reserved.

8.2.2.16.44 DMA Duplicated Good Rx LPBK Packet Counter - RXDLPBKPC (0x00002F74)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GPC	31:0	0x0	RC	NA	Number of replicated or mirrored packets that meet the RXLPBKPC conditions. The sum of RXDLPBKPC and RXLPBKPC is the total good (non-erred) Rx packets from a local VM posted to the host (see Note). Note: The counter might count packets directed to ALL Rx queues or specific Rx queues as defined by the RXDSTATCTRL register.

8.2.2.16.45 DMA Duplicated Good Rx LPBK Byte Counter Low - RXDLPBKBCL (0x00002F78)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GBCL	31:0	0x0	RC	NA	Low 32 bits of the 36 bit Byte Counter of good (non-erred) Rx packets that match the RXDLPBKPC. The counter counts all bytes posted to the host before VLAN strip.

8.2.2.16.46 DMA Duplicated Good Rx LPBK Byte Counter High - RXDLPBKBCH (0x00002F7C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GBCH	3:0	0x0	RC	NA	High 4 bits of the 36 bit Byte Counter associated to RXDLPBKBCL.
RESERVED	31:4	0x0	RSV	NA	Reserved.

8.2.2.16.47 Good Packets Transmitted Count - GPTC (0x00004080)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GPTC	31:0	0x0	RC	NA	Number of good packets transmitted. This register counts good (non-erred) transmitted packets. A good transmit packet is considered one that is 64 or more bytes (from <Destination Address> through <CRC>, inclusively) in length. The register counts transmitted clear packets, secure packets and FC packets.

8.2.2.16.48 Good Octets Transmitted Count Low - GOTCL (0x00004090)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
CNT_L	31:0	0x0	RC	NA	Lower 32 bits of the "Good Octets Transmitted" counter. See complete description in the next register GOTCH.

8.2.2.16.49 Good Octets Transmitted Count High - GOTCH (0x00004094)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
CNT_H	3:0	0x0	RC	NA	Higher 4 bits of the "Good Octets Transmitted" counter. The GOTCL and GOTCH registers make up a logical 36-bit counter of successfully transmitted octets (in packets counted by GPTC). This register includes transmitted bytes in a packet from the <Destination Address> field through the <CRC> field, inclusively.
RESERVED	31:4	0x0	RSV	NA	Reserved.

8.2.2.16.50 DMA Good Tx Packet Counter - TXDGPC (0x000087A0)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GPTC	31:0	0x0	RC	NA	Number of Tx packets from host memory. This counter includes packets that are transmitted to the external network as well as packets that are transmitted only to local VMs. The latter occurs only in VT mode when the local switch is enabled. Dropped packets counted in SSVPC register are not counted here.

8.2.2.16.51 DMA Good Tx Byte Counter Low - TXDGBCL (0x000087A4)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
BCL	31:0	0x0	RC	NA	The low 32 bits of the 36-bit byte counter of the Tx packets that match the TXDGPC. The counter counts all bytes posted by the host and the VLAN (if bytes are added by hardware). Dropped packets counted in SSVPC register are not counted here.

8.2.2.16.52 DMA Good Tx Byte Counter High - TXDGBCH (0x000087A8)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
BCH	3:0	0x0	RC	NA	High 4 bits of the 36-bit byte counter associated to TXDGBCL.
RESERVED	31:4	0x0	RSV	NA	Reserved.

8.2.2.16.53 Receive Undersize Count - RUC (0x000040A4)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RUC	31:0	0x0	RC	NA	Receive Undersize Error. This register counts the number of received frames that are shorter than minimum size (64 bytes from <Destination Address> through <CRC>, inclusively), and had a valid CRC. This register counts packets regardless of L2 filtering and receive enablement.

8.2.2.16.54 Receive Fragment Count - RFC (0x000040A8)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RFC	31:0	0x0	RC	NA	Number of receive fragment errors (frame shorted than 64 bytes from <Destination Address> through <CRC>, inclusively) that have bad CRC (this is slightly different from the Receive Undersize Count register). This register counts packets regardless of L2 filtering and receive enablement.

8.2.2.16.55 Receive Oversize Count - ROC (0x000040AC)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
ROC	31:0	0x0	RC	NA	Receive oversize Error. This register counts the number of received frames that are longer than maximum size as defined by MAXFRS.MFS (from <Destination Address> through <CRC>, inclusively) and have valid CRC. This register counts packets regardless of L2 filtering and receive enablement.

8.2.2.16.56 Receive Jabber Count - RJC (0x000040B0)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RJC	31:0	0x0	RC	NA	Number of receive jabber errors. This register counts the number of received packets regardless of L2 filtering and receive enablement, and are greater than maximum size and have bad CRC (this is slightly different from the Receive Oversize Count register). The packets length is counted from <Destination Address> through <CRC>, inclusively. This register counts packets regardless of L2 filtering and receive enablement.

8.2.2.16.57 Management Packets Received Count - MNGPRC (0x000040B4)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MNGPRC	31:0	0x0	RO	NA	Number of management packets received. This register counts the total number of packets received that pass the management filters. Management packets include RMCP and ARP packets. SFD errors and short packets are not counted, except that packets dropped because the management receives a jumbo packet or because the receive FIFO is full are counted.

8.2.2.16.58 Management Packets Dropped Count - MNGPDC (0x000040B8)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MPDC	31:0	0x0	RO	NA	Number of management packets dropped. This register counts the total number of packets received that pass the management filters and then are dropped because: 1. The management receive FIFO is full or disabled or, 2. a packet bigger than the maximal allowed size is received or, 3. a packet length mismatch. The maximal allowed size is 1518 + the size of any recognized L2 header (VLAN, extended VLAN, E-tag, etc). Management packets include any packet directed to the manageability console (such as RMCP and ARP packets).

8.2.2.16.59 Management Packets Transmitted Count - MNGPTC (0x000041EC)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MPTC	31:0	0x0	RO	NA	Number of management packets transmitted onto the wire.

8.2.2.16.60 Total Octets Received Low - TORL (0x000040C0)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
CNT_L	31:0	0x0	RC	NA	Lower 32 bits of the "Total Octets Received" counter. See complete description in the next register TORH.

8.2.2.16.61 Total Octets Received High - TORH (0x000040C4)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
CNT_H	3:0	0x0	RC	NA	Higher 4 bits of the "Total Octets Received" counter. The TORL and TORH registers make up a logical 36-bit counter of the total received octets (in the packets counted by the TPR counter). This register includes bytes received in a packet from the <Destination Address> field through the <CRC> field, inclusively.
RESERVED	31:4	0x0	RSV	NA	Reserved.

8.2.2.16.62 Total Packets Received - TPR (0x000040D0)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TPR	31:0	0x0	RC	NA	Total Packets Received. All packets received, aside from noted exceptions, are counted in this register regardless of error or L2 filtering decisions. Notes: - This counter doesn't count Flow Control packets. - This counter doesn't count packets that are shorter than 12B or SFD errors as these are terminated in the MAC.

8.2.2.16.63 Total Packets Transmitted - TPT (0x000040D4)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TPT	31:0	0x0	RC	NA	Number of all packets transmitted. This register counts the total number of all packets transmitted. This register counts all packets, including standard packets, secure packets, FC packets, and manageability packets.

8.2.2.16.64 Packets Transmitted (64 Bytes) Count - PTC64 (0x000040D8)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PTC64	31:0	0x0	RC	NA	Number of packets transmitted that are 64 bytes in length (from <Destination Address> through <CRC>, inclusively). This register counts all packets, including standard packets, secure packets, FC packets, and manageability packets.

8.2.2.16.65 Packets Transmitted [65-127 Bytes] Count - PTC127 (0x000040DC)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PTC127	31:0	0x0	RC	NA	Number of packets transmitted that are 65-127 bytes in length (from <Destination Address> through <CRC>, inclusively). This register counts all packets, including standard packets, secure packets, and manageability packets.

8.2.2.16.66 Packets Transmitted [128-255 Bytes] Count - PTC255 (0x000040E0)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PTC255	31:0	0x0	RC	NA	Number of packets transmitted that are 128-255 bytes in length (from <Destination Address> through <CRC>, inclusively). This register counts all packets, including standard packets, secure packets, and manageability packets.

8.2.2.16.67 Packets Transmitted [256-511 Bytes] Count - PTC511 (0x000040E4)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PTC511	31:0	0x0	RC	NA	Number of packets transmitted that are 256-511 bytes in length (from <Destination Address> through <CRC>, inclusively). This register counts all packets, including standard packets, secure packets, and manageability packets.

8.2.2.16.68 Packets Transmitted [512-1023 Bytes] Count - PTC1023 (0x000040E8)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PTC1023	31:0	0x0	RC	NA	Number of packets transmitted that are 512-1023 bytes in length (from <Destination Address> through <CRC>, inclusively). This register counts all packets, including standard packets, secure packets, and manageability packets.

8.2.2.16.69 Packets Transmitted [Greater than 1024 Bytes] Count - PTC1522 (0x000040EC)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PTC1522	31:0	0x0	RC	NA	Number of packets transmitted that are 1024 or more bytes in length (from <Destination Address> through <CRC>, inclusively). This register counts all packets, including standard packets, secure packets, and manageability packets. Due to changes in the standard for maximum frame size for VLAN tagged frames in 802.3, this device transmits packets which have a maximum length of 1522 bytes. The RMON statistics associated with this range has been extended to count 1522 byte long packets. This register counts all packets, including standard and secure packets.

8.2.2.16.70 Multicast Packets Transmitted Count - MPTC (0x000040F0)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MPTC	31:0	0x0	RC	NA	Number of multicast packets transmitted. This register counts the number of multicast packets transmitted. This register counts all packets, including standard packets, secure packets, FC packets and manageability packets.

8.2.2.16.71 Broadcast Packets Transmitted Count - BPTC (0x000040F4)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
BPTC	31:0	0x0	RC	NA	Number of broadcast packets transmitted count. This register counts all packets, including standard packets, secure packets, FC packets and manageability packets

8.2.2.16.72 MAC short Packet Discard Count - MSPDC (0x00004010)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MSPDC	31:0	0x0	RC	NA	Number of MAC short Packet Discard packets received.

8.2.2.16.73 XSUM Error Count - XEC (0x00004120)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
XEC	31:0	0x0	RC	NA	Number of Receive IPv4, TCP, UDP or SCTP XSUM Errors.

Note: XSUM errors are not counted when a packet has any MAC error (CRC, length, under-size, over-size, byte error or symbol error).

8.2.2.16.74 Receive Queue Statistic Mapping Registers - RQSMR[n] (0x00002300 + 0x4*n, n=0...31)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
Q_MAP_0	3:0	0x0	RW	NA	For each register 'n', Q_MAP[0] defines the per queue statistic registers that are mapped to Rx queue '4*n+0'. (see examples below)
RESERVED	7:4	0x0	RSV	NA	Reserved.
Q_MAP_1	11:8	0x0	RW	NA	For each register 'n', Q_MAP[1] defines the per queue statistic registers that are mapped to Rx queue '4*n+1'. (see examples below)
RESERVED	15:12	0x0	RSV	NA	Reserved.
Q_MAP_2	19:16	0x0	RW	NA	For each register 'n', Q_MAP[2] defines the per queue statistic registers that are mapped to Rx queue '4*n+2'. (see examples below)
RESERVED	23:20	0x0	RSV	NA	Reserved.
Q_MAP_3	27:24	0x0	RW	NA	For each register 'n', Q_MAP[3] defines the per queue statistic registers that are mapped to Rx queue '4*n+3'. (see examples below)
RESERVED	31:28	0x0	RSV	NA	Reserved.

8.2.2.16.75 Rx DMA Statistic Counter Control - RXDSTATCTRL (0x00002F40)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
QSEL	4:0	0x0	RW	NA	The Queue Select field controls which Rx queues are considered for the DMA Good Rx and DMA Duplicated counters as follow: 00000...01111: The counters relates to the same queues that are directed to the QPRC[QSEL] counter as defined by the RQSMR[n] registers. 10000: The counters relates to all Rx queues. Else: Reserved.
RESERVED	31:5	0x0	RSV	NA	Reserved.

8.2.2.16.76 Transmit Queue Statistic Mapping Registers - TQSM[n] (0x00008600 + 0x4*n, n=0...31)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
Q_MAP_0	3:0	0x0	RW	NA	For each register 'n', Q_MAP[0] defines the per queue statistic registers that are mapped to Tx queue '4*n+0'.
RESERVED	7:4	0x0	RSV	NA	Reserved.
Q_MAP_1	11:8	0x0	RW	NA	For each register 'n', Q_MAP[1] defines the per queue statistic registers that are mapped to Tx queue '4*n+1'.
RESERVED	15:12	0x0	RSV	NA	Reserved.
Q_MAP_2	19:16	0x0	RW	NA	For each register 'n', Q_MAP[2] defines the per queue statistic registers that are mapped to Tx queue '4*n+2'.
RESERVED	23:20	0x0	RSV	NA	Reserved.
Q_MAP_3	27:24	0x0	RW	NA	For each register 'n', Q_MAP[3] defines the per queue statistic registers that are mapped to Tx queue '4*n+3'.
RESERVED	31:28	0x0	RSV	NA	Reserved.

8.2.2.16.77 Queue Packets Received Count - QPRC[n] (0x00001030 + 0x40*n, n=0...15)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PRC	31:0	0x0	RC	NA	Number of packets received for the Queue.

8.2.2.16.78 Queue Packets Received Drop Count - QPRDC[n] (0x00001430 + 0x40*n, n=0...15)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PRDC	31:0	0x0	RC	NA	Number of receive packets dropped for the queue. Packets are dropped per queue in one of three cases: 1. Rx Queue is disabled in the RXDCTL[n] register (this includes packets sent to this queue due to the flow control drop no match mechanism or drop action). 2. No free descriptors in the Rx queue while hardware is set to Drop En in the SRRCTL[n] register or in the PFQDE register. 3. Packet size is larger than RLPML while RLPML_EN is set in the RXDCTL[n] register.

8.2.2.16.79 Queue Bytes Received Count Low - QBRC_L[n] (0x00001034 + 0x40*n, n=0...15)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
BRC_L	31:0	0x0	RC	NA	Lower 32 bits of the statistic counter. The QBRC_L[n] and QBRC_H[n] registers make up a logical 36-bit counter of received bytes that were posted to the programmed Rx queues of the packets counted by the QPRC[n]. The counter counts all bytes posted to the host before VLAN strip.

8.2.2.16.80 Queue Bytes Received Count High - QBRC_H[n] (0x00001038 + 0x40*n, n=0...15)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
BRC_H	3:0	0x0	RC	NA	Higher 4 bits of the statistic counter described in QBRC_L.
RESERVED	31:4	0x0	RSV	NA	Reserved.

8.2.2.16.81 Queue Packets Transmitted Count - QPTC[n] (0x00008680 + 0x4*n, n=0...15)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PTC	31:0	0x0	RC	NA	Number of packets transmitted for the Queue. A packet is considered as transmitted if it was forwarded to the MAC unit for transmission to the network and/or is accepted by the internal Tx to Rx switch enablement logic. Packets dropped due to anti-spoofing filtering, or traffic sent to the BMC and blocked due to security violations, or loopback packets that are rejected by the Tx to Rx switch, are not counted.

Note: Additional address(es): 0x06030 + 0x40*n, n=0...15

8.2.2.16.82 Queue Packets Transmitted Count Alias - QPTC_ALIAS[n] (0x00006030 + 0x40*n, n=0...15; RC)

Reset source: Master Reset, NVM Load: No, Scope: PF

Fields definitions are the same as defined on [Section 8.2.2.16.81](#)

8.2.2.16.83 Queue Bytes Transmitted Count Low - QBTC_L[n] (0x00008700 + 0x8*n, n=0...15)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
BTC_L	31:0	0x0	RC	NA	Lower 32 bits of the statistic counter. The QBTC_L and QBTC_H registers make up a logical 36-bit counter of transmitted bytes of the packets counted by the matched QPTC counter. These registers count all bytes in the packets from the <Destination Address> field through the <CRC> field, inclusively. These registers must be accessed as two consecutive 32bit entities while QBTC_L register is read first, or a single 64bit read cycle. Each register is Read cleared. In addition, it sticks at 0xFF...F to avoid overflow.

8.2.2.16.84 Queue Bytes Transmitted Count High - QBTC_H[n] (0x00008704 + 0x8*n, n=0...15)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
BTC_H	3:0	0x0	RC	NA	Higher 4 bits of the statistic counter described in QBTC_L.
RESERVED	31:4	0x0	RSV	NA	Reserved.

8.2.2.16.85 Fiber Channel CRC Error Count - FCCRC (0x00005118)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
CRC_CNT	15:0	0x0	RC	NA	FC CRC Count. Count the number of packets with good Ethernet CRC and bad FC CRC.
RESERVED	31:16	X	RSV	NA	Reserved.

8.2.2.16.86 Fiber Channel Last Error Count - FCLAST (0x00002424)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	15:0				Reserved
RESERVED	31:16	0x0	RSV	NA	Reserved.

8.2.2.16.87 BMC2OS Packets Sent by BMC - B2OSPC (0x000041C0)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
B2OSPC	31:0	0x0	RC	NA	BMC2OS packets sent by BMC.

8.2.2.16.88 BMC2OS Packets Received by Host - B2OGPRC (0x00002F90)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
B2OGPRC	31:0	0x0	RC	NA	BMC2OS packets received by host.

8.2.2.16.89 OS2BMC packets received by BMC - O2BGPTC (0x000041C4)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
O2BGPTC	31:0	0x0	RC	NA	OS2BMC good packets transmitted count. This includes packets sent from the host to the BMC or consumed by the internal manageability engine.

8.2.2.16.90 OS2BMC packets transmitted by host - O2BSPC (0x000087B0)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
O2BPC	31:0	0x0	RC	NA	OS2BMC packets transmitted count.

8.2.2.16.91 Total Unicast Packets Received (BMC copy) - BUPRC (0x00004180)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
BUPRC	31:0	0x0	RC	NA	Number of Unicast packets received.,

8.2.2.16.92 BMC Total Multicast Packets Received - BMPRC (0x00004184)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
BMPRC	31:0	0x0	RC	NA	Number of good (non-erred) multicast packets received that pass L2 filtering (excluding Broadcast packets). This register does not count received flow control packets. This registers counts packets regardless on receive enablement.

8.2.2.16.93 Total Broadcast Packets Received (BMC copy) - BBPRC (0x00004188)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
BBPRC	31:0	0x0	RC	NA	Number of good (non-erred) broadcast packets received to BMC. This register does not count received broadcast packets when the broadcast address filter is disabled. The counter counts packets regardless on receive enablement.

8.2.2.16.94 Total Unicast Packets Transmitted (BMC copy) - BUPTC (0x0000418C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
BUPTC	31:0	0x0	RC	NA	Number of unicast packets transmitted.

8.2.2.16.95 BMC Total Multicast Packets Transmitted - BMPTC (0x00004190)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
BMPTC	31:0	0x0	RC	NA	Number of multicast packets transmitted. This register counts the number of multicast packets transmitted. This register counts all packets, including standard packets, secure packets, FC packets and manageability packets.

8.2.2.16.96 Total Broadcast Packets Transmitted (BMC copy) - BBPTC (0x00004194)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
BBPTC	31:0	0x0	RC	NA	Number of broadcast packets transmitted count. This register counts all packets, including standard packets, secure packets, FC packets and manageability packets

8.2.2.16.97 BMC FCS Receive Errors - BCRERRS (0x00004198)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
BCEC	31:0	0x0	RC	NA	CRC error count. Counts the number of receive packets with CRC errors. In order for a packet to be counted in this register, it must be 64 bytes or greater (from <Destination Address> through <CRC>, inclusively) in length. This registers counts all packets received, regardless of L2 filtering and receive enablement.

8.2.2.16.98 BMC Pause XON Frames Received - BXONRXC (0x0000419C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
BXONRXC	15:0	0x0	RC	NA	Number of XON packets Received. Sticks to 0xFFFF. XON packets can use the global address, or the station address. This register counts any XON packet whether it is a legacy XON or a priority XON. Each XON packet is counted once even if it designated to a few priorities. If a priority FC packet contains both XOFF and XON, only the LXOFFRXCNT counter is incremented.
RESERVED	31:16	0x0	RC	NA	Reserved

8.2.2.16.99 BMC Pause XOFF Frames Received - BXOFFRXC (0x000041E0)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
BXOFFRXC	15:0	0x0	RC	NA	Number of XOFF packets Received. Sticks to 0xFFFF. XOFF packets can use the global address, or the station address. This register counts any XOFF packet whether it is a legacy XOFF or a priority XOFF. Each XOFF packet is counted once even if it designated to a few priorities. If a priority FC packet contains both XOFF and XON, only this counter is incremented.
RESERVED	31:16	0x0	RC	NA	Reserved

8.2.2.16.100 BMC Pause XON Frames Transmitted - BXONTXC (0x000041E4)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
BXONTXC	15:0	0x0	RC	NA	Number of XON packets Transmitted. Sticks to 0xFFFF. XONTXC is incremented by one for each Link XON packet when MFLCN.RFCE is set and for each Priority XON packet when corresponding MFLCN.RPFCE bit is set.
RESERVED	31:16	0x0	RC	NA	Reserved.

8.2.2.16.101 BMC Pause XOFF Frames Transmitted - BXOFFTXC (0x000041E8)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
BXOFFTXC	15:0	0x0	RC	NA	Number of XOFF packets Transmitted. Sticks to 0xFFFF. XOFFTXC is incremented by one for each Link XOFF packet when MFLCN.RFCE is set and for each Priority XOFF packet when corresponding MFLCN.RPFCE bit is set.
RESERVED	31:16	0x0	RC	NA	Reserved.

8.2.2.16.102 Sideband receive dropped packet count - B2OSDPC (0x000041F0)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
B2OSDPC	31:0	0x0	RW	NA	Counts the number of packets received from the BMC interface receive that where dropped.

8.2.2.17 Wake-Up Control Registers

8.2.2.17.1 Wake Up Control Register - WUC (0x00005800)

Reset source: LAN Power Good Reset, NVM Load: Dedicated, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	0	0b	RSV	NA	Reserved.
PME_EN	1	0b	RW	NA	This bit is used by the software device driver to read the PME_En bit of the Power Management Control/Status Register (PMCSR) without accessing the PCIe configuration space. Note: Software should not modify this bit while PME Enablement is active.
PME_STATUS	2	0b	RW1C	DYNAMIC	This bit is set when the device receives a wake-up event. It is the same as the PME_Status bit in the Power Management Control/Status Register (PMCSR). Writing a 1b to this bit clears it together with the PME_Status bit in the PMCSR.
RESERVED	3	0b	RSV	NA	Reserved.
WKEN	4	1b	RW	NA	This bit can be cleared to disable the PE_WAKE_N pin assertion (= 0) even if APM is enabled in the NVM. In this case, PMCSR and WUS wake-up statuses are invalid. Note: This bit should not be cleared while in ACPI mode.
RESERVED	31:5	0x0	RSV	NA	Reserved.

Note: The PME_En and PME_Status bits are reset when LAN_PWR_GOOD is 0. When AUX_PWR=0, these bits are also reset by the assertion of PE_RST_N.

8.2.2.17.2 Wake Up Filter Control Register - WUFC (0x00005808)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
LNKC	0	0b	RW	NA	Link Status Change Wake Up Enable.
MAG	1	0b	RW	NA	Magic Packet Wake Up Enable.
EX	2	0b	RW	NA	Directed Exact Wake Up Enable.
MC	3	0b	RW	NA	Directed Multicast Wake Up Enable. Setting this bit does not enable broadcast packets, which are enabled by the BC bit in this register.
BC	4	0b	RW	NA	Broadcast Wake Up Enable.
ARP	5	0b	RW	NA	ARP/IPv4 Request Packet Wake Up Enable.
IPv4	6	0b	RW	NA	Directed IPv4 Packet Wake Up Enable.
IPv6	7	0b	RW	NA	Directed IPv6 Packet Wake Up Enable.
RESERVED	14:8	0x0	RSV	NA	Reserved.
NOTCO	15	0b	RW	NA	Ignore TCO/managements packets for wake up. 0b = When set to 0 packet that are marked to be sent both to MNG and Host will cause WoL. 1b = When set to 1, packets that are marked to be forwarded both to MNG and Host don't wake up the CPU.

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FLX0	16	0b	RW	NA	Flexible Filter 0 Enable: Controls the usage of the FHFT[0] register (0x9000).
FLX1	17	0b	RW	NA	Flexible Filter 1 Enable. Controls the usage of the FHFT[1] register (0x9100).
FLX2	18	0b	RW	NA	Flexible Filter 2 Enable. Controls the usage of the FHFT[2] register (0x9200).
FLX3	19	0b	RW	NA	Flexible Filter 3 Enable. Controls the usage of the FHFT[3] register (0x9300).
FLX4	20	0b	RW	NA	Flexible Filter 4 Enable. Controls the usage of the FHFT[0] register (0x9600).
FLX5	21	0b	RW	NA	Flexible Filter 5 Enable. Controls the usage of the FHFT[5] register (0x9700).
FLX6	22	0b	RW	NA	Flexible Filter 6 Enable. Controls the usage of the FHFT[6] register (0x9800).
FLX7	23	0b	RW	NA	Flexible Filter 7 Enable. Controls the usage of the FHFT[7] register (0x9900).
RESERVED	30:24	0x0	RSV	NA	Reserved.
FW_RST_WK	31	0b	RW	NA	Enable Wake on Firmware Reset Assertion. When set, a firmware reset causes a system wake up that enables the software driver to resend proxying information to firmware.

Note: This register is used to enable each of the pre-defined and flexible filters for wake up support. A value of one means the filter is turned on, and a value of zero means the filter is turned off.

8.2.2.17.3 Wake Up Status Register - WUS (0x00005810)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
LNKC	0	0b	RW1C	DYNAMIC	Link Status Changed. This bit can be set even if another event is already set.
MAG	1	0b	RW1C	DYNAMIC	Magic Packet Received.
EX	2	0b	RW1C	DYNAMIC	Directed Exact Packet Received. The packet's address matched one of the 16 pre-programmed exact values in the Receive Address registers.
MC	3	0b	RW1C	DYNAMIC	Directed Multicast Packet Received. The packet was a multicast packet that's hashed to a value that corresponds to a 1 bit in the Multicast Table Array.
BC	4	0b	RW1C	DYNAMIC	Broadcast Packet Received.
ARP	5	0b	RW1C	DYNAMIC	ARP/IPv4 Request Packet Received.
IPV4	6	0b	RW1C	DYNAMIC	Directed IPv4 Packet Received.
IPV6	7	0b	RW1C	DYNAMIC	Directed IPv6 Packet Received.
MNG	8	0b	RW1C	DYNAMIC	Indicates that a manageability event that should cause a PME to happen.
RESERVED	15:9	0x0	RSV	NA	Reserved.
FLX0	16	0b	RW1C	DYNAMIC	Flexible Filter 0 Match.
FLX1	17	0b	RW1C	DYNAMIC	Flexible Filter 1 Match.
FLX2	18	0b	RW1C	DYNAMIC	Flexible Filter 2 Match.
FLX3	19	0b	RW1C	DYNAMIC	Flexible Filter 3 Match.
FLX4	20	0b	RW1C	DYNAMIC	Flexible Filter 4 Match.
FLX5	21	0b	RW1C	DYNAMIC	Flexible Filter 5 Match.

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FLX6	22	0b	RW1C	DYNAMIC	Flexible Filter 6 Match.
FLX7	23	0b	RW1C	DYNAMIC	Flexible Filter 7 Match.
RESERVED	30:24	0x0	RSV	NA	Reserved.
FW_RST_WK	31	0b	RW1C	DYNAMIC	Wake Due to Firmware Reset Assertion Event. When set to 1b, indicates that asserting a firmware reset causes the system wake up so the software driver can re-send proxying information to firmware. This bit can be set even if another event is already set.

Notes:

- This register is valid only when PMCSR.PME_en is set to 1.
- Software writing a 1b to any bit clears that bit.
- This register is not cleared when PE_RST_N is asserted. It is only cleared when LAN_PWR_GOOD is de-asserted or when cleared by the driver.

8.2.2.17.4 IP Address Valid - IPAV (0x00005838)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
V40	0	0b	RW	NA	IPv4 Address 0 Valid Loaded from NVM
V41	1	0b	RW	NA	IPv4 Address 1 Valid
V42	2	0b	RW	NA	IPv4 Address 2 Valid
V43	3	0b	RW	NA	IPv4 Address 3 Valid
RESERVED	15:4	0x0	RSV	NA	Reserved.
V60	16	0b	RW	NA	IPv6 Address 0 Valid
V61	17	0b	RW	NA	IPv6 Address 1 Valid
V62	18	0b	RW	NA	IPv6 Address 2 Valid
V63	19	0b	RW	NA	IPv6 Address 3 Valid
RESERVED	31:20	0x0	RSV	NA	Reserved.

8.2.2.17.5 IPv4 Address Table - IP4AT[n] (0x00005840 + 0x8*n, n=0...3)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
IPV4ADDR	31:0	X	RW	NA	IPv4 Address 'n', 'n' = 0...3.

8.2.2.17.6 IPv6 Address Table Extended - IP6AT_EXT[n] (0x00005990 + 0x4*n, n=0...11)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
IPV6ADDR	31:0	X	RW	NA	4 x Register IPv6 filter. Register 'n' contains bytes '4*n' up to '4*n+3' of the IPv6 address. LS byte of register '0' is first on the wire.

8.2.2.17.7 IPv6 Address Table - IP6AT[n] (0x00005880 + 0x4*n, n=0...3)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
IPV6ADDR	31:0	X	RW	NA	4 x Register IPv6 filter. Register 'n' contains bytes '4*n' up to '4*n+3' of the IPv6 address. LS byte of register '0' is first on the wire.

8.2.2.17.8 Wake Up Packet Length - WUPL (0x00005900)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
LEN	15:0	X	RO	NA	Length of Wake Up Packet. This bit indicates the length of the first wake-up packet received. It is valid if one of the bits in the Wake Up Status Register (WUS) is set. It is not cleared by any reset.
RESERVED	31:16	0x0	RSV	NA	Reserved.

8.2.2.17.9 Wake Up Packet Memory (128 Bytes) - WUPM[n] (0x00005A00 + 0x4*n, n=0...31)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
WUPD	31:0	X	RO	NA	Wake Up Packet Data. This register is used to store the first 128 bytes of the wake up packet for software retrieval after the system wakes up. It should not be cleared by any reset including master reset.

8.2.2.17.10 Filter DW Even - FHFT_FILTER_DW_EVEN[n,m] (0x00009000 + 0x10*n + 0x100*m, n=0...15, m=0...3 and 0x00009600 + 0x10*(n-16) + 0x100*m, n=16...31, m=0...3)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FHFT_FILTER_0_DW0	31:0	X	RW	NA	Even DW of the flex filter

8.2.2.17.11 Filter DW Odd - FHFT_FILTER_DW_ODD[n,m] (0x00009604 + 0x10*n + 0x100*m, n=0...15, m=0...3 and 0x00009004 + 0x10*(n-16) + 0x100*m, n=16...31, m=0...3)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FHFT_FILTER_0_DW1	31:0	X	RW	NA	Odd DW of the flex filter

8.2.2.17.12 Filter Mask - FHFT_FILTER_MASK[n,m] (0x00009608 + 0x10*n + 0x100*m, n=0...15, m=0...3 and 0x00009008 + 0x10*(n-16) + 0x100*m, n=16...31, m=0...3)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FHFT_FILTER_0_MASK	7:0	X	RW	NA	Bit mask for the 8 bytes of the filter (bit per byte)
RESERVED	31:8	0x0	RSV	NA	

8.2.2.17.13 Filter Length - FHFT_FILTER_LENGTH[n,m] (0x0000900C + 0x10*n + 0x100*m, n=0...15, m=0...3 and 0x0000960C + 0x10*(n-16) + 0x100*m, n=16...31, m=0...3)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
LENGTH	7:0	0x0	RW	NA	Flex filter length The length field is only valid on the last DW of the filter, all other length fields are reserved
RESERVED	31:8	0x0	RSV	NA	

8.2.2.18 Management Filters Registers

The Management Filters registers are RO for the host. These registers are initialized at LAN Power Good and can be loaded from the NVM by the manageability firmware.

8.2.2.18.1 Manageability Only Traffic - MNGONLY (0x00005864)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
EXCLUSIVE_T O_MNG	7:0	0x0	RW	NA	Exclusive to MNG - when set, indicates that packets forwarded by the manageability filters to manageability are not sent to the host. Bits 0...7 correspond to decision rules defined in registers MDEF[0...7] and MDEF_EXT[0...7].
RESERVED	31:8	0x0	RSV	NA	Reserved

8.2.2.18.2 Management VLAN TAG Value - MAVTV[n] (0x00005010 + 0x4*n, n=0...7)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VID	11:0	0x0	RW	NA	Contain the VLAN ID that should be compared with the incoming packet if the corresponding bit in MFVAL.VLAN is set.
RESERVED	31:12	0x0	RSV	NA	Reserved.

8.2.2.18.3 Management Flex UDP/TCP Ports - MFUTP[n] (0x00005030 + 0x4*n, n=0...7)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MFUTP_2N	15:0	0x0	RW	NA	(2n)-th Management Flex UDP/TCP port
MFUTP_2N_1	31:16	0x0	RW	NA	(2n+1)-th Management Flex UDP/TCP port

Notes:

- Each 32-bit register (n=0,...,7) refers to two port filters (register 0 refers to ports 0&1, register 2 refers to ports 2&3, etc.).
- SCTP packets do not match the MFUTP filters.
- MFUTP filters are programmed in Network order.

8.2.2.18.4 Management Ethernet Type Filters - METF[n] (0x00005190 + 0x4*n, n=0...3)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
ETYPE	15:0	0x0	RW	NA	EtherType value to be compared against the L2 EtherType field in the Rx packet. Note: Appears in Little Endian order (high byte first on the wire).
RESERVED	29:16	0x0	RSV	NA	Reserved.
POLARITY	30	0b	RW	NA	0 = Positive filter - Filter enters the decision filters if a match occurred. 1 = Negative filter - Filter enters the decision filters if a match did not occur.
RESERVED	31	0b	RSV	NA	Reserved.

8.2.2.18.5 Management Control Register - MANC (0x00005820)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FC_DISCARD	0	0b	RW	NA	0 = Apply filtering rules to packets with Flow Control EtherType. 1 = Discard packets with Flow Control EtherType. Note: Flow Control EtherType is 0x8808
NCSI_DISCARD	1	0b	RW	NA	0 = Apply filtering rules to packets with NC-SI EtherType. 1 = Discard packets with NC-SI EtherType. Note: NC-SI EtherType is 0x88F8
RESERVED	16:2	0x0	RSV	NA	Reserved.
RCV_TCO_EN	17	0b	RW	NA	Receive TCO Packets Enabled. When this bit is set it enables the receive flow to the manageability block. This bit should be set only if at least one of MANC.EN_BMC2OS or MANC.EN_BMC2NET bits are set.
RESERVED	18	0b	RSV	NA	Reserved.
RCV_ALL	19	0b	RW	NA	Receive All Enable. When set, all packets are received from the wire and passed to the manageability block.
RESERVED	22:20	0x0	RW	NA	Reserved
EN_XSUM_FILTER	23	0b	RW	NA	When set this bit enables Xsum filtering to Manageability. Meaning, only packets that pass L3, L4 checksum are sent to the manageability block. Note: This capability is not provided for tunneled packets. This bit should be set in functional mode to enable Checksum.
EN_IPV4_FILTER	24	0b	RW	NA	Enable IPv4 address Filters - when set, the last 128 bits of the MIPAF register are used to store 4 IPv4 addresses for IPv4 filtering. When cleared, these bits store a single IPv6 filter.
FIXED_NET_TYPE	25	0b	RW	NA	Fixed next type: If set, only packets matching the net type defined by the NET_TYPE field will pass to manageability. Otherwise, both tagged and untagged packets might be forwarded to manageability engine.

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
NET_TYPE	26	0b	RW	NA	NET TYPE: 0: pass only un-tagged packets. 1: pass only VLAN tagged packets. Valid only if FIXED_NET_TYPE is set.
RESERVED	27				Reserved
EN_BMC2OS	28	0b	RW	NA	Enable BMC2OS and OS2BMC traffic 0 = The BMC can not communicate with the OS. 1 = The BMC can communicate with the OS. When cleared the BMC traffic is not forwarded to the OS, even if the Host MAC address filter and VLANs (RAH/L, MTA, VFTA and PFVLVF registers) indicate that it should. When cleared the OS traffic is not forwarded to the BMC even if the decision filters indicates it should. This bit does not impact the BMC to Network traffic. Note: This bit can change while the host is sending or receiving traffic.
EN_BMC2NET	29	0b	RW	NA	Enable BMC to network and network to BMC traffic 0 = The BMC can not communicate with the network. 1 = The BMC can communicate with the network When cleared the BMC traffic is not forwarded to the network and the network traffic is not forwarded to the BMC even if the decision filters indicates it should. This bit does not impact the host to BMC traffic. Note: This bit can change while the host is sending or receiving traffic.
RESERVED	31:30	00b	RSV	NA	Reserved.

8.2.2.18.6 Manageability Decision Filters - MDEF[n] (0x00005890 + 0x4*n, n=0...7)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
EXACT_AND	3:0	0x0	RW	NA	Exact - Controls the inclusion of Exact MAC address 0 to 3 In the manageability filter decision (AND section). Bit 0 corresponds to exact MAC address 0 (MMAL0 and MMAH0), etc.
BROADCAST_AND	4	0b	RW	NA	Broadcast - Controls the inclusion of broadcast address filtering in the manageability filter decision (AND section).
VLAN_AND	12:5	0x0	RW	NA	VLAN - Controls the inclusion of VLAN tag 0 to 7 respectively In the manageability filter decision (AND section). Bit 5 corresponds to VLAN tag 0, etc.
IPV4_ADDRESSES	16:13	0x0	RW	NA	IPv4 Address - Controls the inclusion of IPV4 address 0 to 3 (MIPAF[3,n]) respectively in the manageability filter decision (AND section). Bit 13 corresponds to IPV4 address 0, etc. Notes: - This field is relevant only if MANC.EN_IPv4_FILTER is set. - Supported only for Network traffic.

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
IPV6_ADDRESSES	20:17	0x0	RW	NA	IPv6 Address - Controls the inclusion of IPv6 address 0 to 3 respectively (MIPAF[0:3,n]) in the manageability filter decision (AND section). Bit 17 corresponds to IPV6 address 0, etc Notes: • Bit 20 is relevant only if MANC.EN_IPv4_FILTER is cleared. • Supported only for Network traffic.
EXACT_OR	24:21	0x0	RW	NA	Exact - Controls the inclusion of exact MAC address 0 to 3 In the manageability filter decision (OR section). Bit 21 corresponds to exact MAC address 0 (MMAL0 and MMAH0), etc.
BROADCAST_OR	25	0b	RW	NA	Broadcast - Controls the inclusion of broadcast address filtering in the manageability filter decision (OR section).
MULTICAST_AND	26	0b	RW	NA	Multicast - Controls the inclusion of Multicast address filtering in the manageability filter decision (AND section). Broadcast packets are not included by this bit.
ARP_REQUEST	27	0b	RW	NA	ARP Request - Controls the inclusion of ARP Request filtering in the manageability filter decision (OR section). Note: Supported only for Network traffic.
ARP_RESPONSE	28	0b	RW	NA	ARP Response - Controls the inclusion of ARP Response filtering in the manageability filter decision (OR section). Note: Supported only for Network traffic.
ND_134	29	0b	RW	NA	Neighbor Discovery - Controls the inclusion of Router Advertisement neighbor Discovery filtering in the manageability filter decision (OR section). Notes: 1. Supported only for Network traffic. 2. Neighbor Discovery types supported by this bit is 0x86 (134d) - Router Advertisement
PORT_0X298	30	0b	RW	NA	Port 0x298 - Controls the inclusion of Port 0x298 filtering in the manageability filter decision (OR section). Note: Supported only for Network traffic.
PORT_0X26F	31	0b	RW	NA	Port 0x26F - Controls the inclusion of Port 0x26F filtering in the manageability filter decision (OR section). Note: Supported only for Network traffic.

8.2.2.18.7 Manageability Decision Filters Ext - MDEF_EXT[n] (0x00005160 + 0x4*n, n=0...7)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
L2_ETHERTYPE_AND	3:0	0x0	RW	NA	L2 EtherType: Controls the inclusion of L2 EtherType filtering in the manageability filter decision (AND section).
L2_ETHERTYPE_OR	7:4	0x0	RW	NA	L2 EtherType: Controls the inclusion of L2 EtherType filtering in the manageability filter decision (OR section).
FLEX_PORT	23:8	0x0	RW	NA	Flex port - Controls the inclusion of Flex port filtering in the manageability filter decision (OR section). Bit 8 corresponds to flex port 0, etc.
FLEX_TCO	24	0b	RW	NA	Flex TCO - Controls the inclusion of Flex TCO filtering in the manageability filter decision (OR section). Bit 24 corresponds to Flex TCO filter 0. Note: Supported only for Network traffic.
ND_135	25	0b	RW	NA	Neighbor Discovery - Controls the inclusion of Neighbor Solicitation neighbor Discovery filtering in the manageability filter decision (OR section). Notes: - Supported only for Network traffic. - Neighbor Discovery types supported by this bit is 0x87 (135d) - Neighbor Solicitation
ND_136	26	0b	RW	NA	neighbor Discovery - Controls the inclusion of Neighbor Advertisement Neighbor Discovery filtering in the manageability filter decision (OR section). Notes: - Supported only for Network traffic. - Neighbor Discovery types supported by this bit is 0x88 (136d) - Neighbor Advertisement
ND_137	27	0b	RW	NA	Neighbor Discovery - Controls the inclusion of Redirect neighbor Discovery filtering in the manageability filter decision (OR section). Notes: - Supported only for Network traffic. - Neighbor Discovery types supported by this bit is 0x89 (137d) - Redirect
RESERVED	28	0b	RSV	NA	
MLD	29	0b	RW	NA	MLD - control the inclusion of MLD packets. These are ICMPv6 packets with the following types: 130, 131, 132, 143.
APPLY_TO_NETWORK_TRAFFIC	30	0b	RW	NA	0= Do not apply this decision filter to traffic received from the network. 1= Apply this decision filter to traffic received from the network.
APPLY_TO_HOST_TRAFFIC	31	0b	RW	NA	0 = This decision filter does not apply to traffic received from the host. 1 = This decision filter applies to traffic received from the host.

8.2.2.18.8 BMC IP address Register - BMCIP[n] (0x00005050 + 0x4*n, n=0...3)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
IPADDR	31:0	0x0	RW	NA	4 bytes of 16 bytes destination IP address of the BMC. n=0 contains the MSB for an IPv6 IP address. n=3 contains an IPv4 IP address or the LSB for an IPv6 IP address. For an IPv4 address, BMCIP 0...2 shall be written with zeros. Note: Field is defined in Big Endian (LS byte is first on the wire).

8.2.2.18.9 BMC IP Valid Register - BMCIPVAL (0x00005060)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
IPADDR_TYPE	0	0b	RW	NA	0 = IPv4 1 = IPv6.
IPADDR_VALID	1	0b	RW	NA	0 = IP address in BMCIP is not valid 1 = IP address in BMCIP is valid
RESERVED	31:2	0x0	RSV	NA	Reserved.

8.2.2.18.10 Manageability IP Address Filter - MIPAF[n,m] (0x000058B0 + 0x4*n + 0x10*m, n=0...3, m=0...3)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
IP_ADDR	31:0	X	RW	NA	Manageability IP Address Filters. For each n, m, m=0...3, n=0...3 while MANC.EN_IPv4_FILTER = 0, MIPAF[m,n] register holds DW 'n' of IPv6 filter 'm' (4 x IPv6 filters). For each n, m, m=0...3, n=0...3 while MANC.EN_IPv4_FILTER = 1, MIPAF[m,n] registers for m=0,1,2 is the same as the previous case (3 x IPv6 filters). And MIPAF[3,n] registers holds IPv4 filter 'n' (4 x IPv4 filters). Note: These registers appear in Big Endian order (LS byte, LS address is first on the wire).

8.2.2.18.11 Manageability Ethernet MAC Address Low - MMAL[n] (0x00005910 + 0x8*n, n=0...3)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MMAL	31:0	X	RW	NA	Manageability Ethernet MAC Address Low. The lower 32 bits of the 48 bit Ethernet MAC Address. Note: Appear in Big Endian order (LS byte of MMAL is first on the wire).

8.2.2.18.12 Manageability Ethernet MAC Address High - MMAH[n] (0x00005914 + 0x8*n, n=0...3)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MMAH	15:0	X	RW	NA	Manageability Ethernet MAC Address High. The upper 16 bits of the 48 bit Ethernet MAC Address. Note: Appear in Big Endian order (MS byte of MMAH is last on the wire).
RESERVED	31:16	0x0	RSV	NA	Reserved. Reads as 0. Ignored on write.

8.2.2.18.13 FTFT Filter DW Even words - FTFT_FILTER_EVEN[n] (0x00009400 + 0x10*n, n=0...15)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FTFT_FILTER_0_DW0	31:0	X	RW	NA	Even DW filter value

Notes:

- The mask field must be 8 bytes aligned even if the length field is not 8 bytes aligned as hardware implementation compares 8 bytes at a time so it should get extra masks until the end of the next quad word. Any mask bit that is located after the length should be set to 0 indicating no comparison should be done.
- In case the actual length which is defined by the length field register and the mask bits is not 8 bytes aligned there might be a case that a packet which is shorter then the actual required length passes the flexible filter. This might happen due to comparison of up to 7 bytes that come after the packet but are not a real part of the packet.
- The last DW of each filter contains a length field defining the number of bytes from the beginning of the packet compared by this filter. If actual packet length is less than length specified by this field, the filter fails. Otherwise, it depends on the result of actual byte comparison. The value should not be greater than 128.
- FTFT registers are configured by firmware.

8.2.2.18.14 FTFT Filter DW Odd words - FTFT_FILTER_ODD[n] (0x00009404 + 0x10*n, n=0...15)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FTFT_FILTER_0_DW1	31:0	X	RW	NA	Odd DW filter value

8.2.2.18.15 FTFT Filter Mask - FTFT_FILTER0_MASK[n] (0x00009408 + 0x10*n, n=0...15)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FTFT_FILTER_0_MASK	7:0	X	RW	NA	Mask for the 8 bytes of the filter (bit per byte)
RESERVED	31:8	0x0	RSV	NA	

8.2.2.18.16 FTFT Filter Length - FTFT_FILTER0_LENGTH[n] (0x0000940C + 0x10*n, n=0...15)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
LENGTH	7:0	0x0	RW	NA	Length field is valid only on the last DW of the Filter all other fields are reserved
RESERVED	31:8	0x0	RSV	NA	

8.2.2.19 Time Sync (IEEE 1588) Registers

8.2.2.19.1 System time register Low - SYSTIMEL (0x00008C0C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
STL	29:0	0x0	RW	NA	System time LSB register, defined in nsec. Writes of values above a second (above 0x3B9AC9FF) are ignored
RESERVED	31:30	0x0	RW	NA	Reserved

8.2.2.19.2 System time register High - SYSTIMEH (0x00008C10)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
STH	31:0	0x0	RW	NA	System time MSB register, defined in seconds

8.2.2.19.3 System time register residue - SYSTIMR (0x00008C58)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
SYSTIMR	31:0	0x0	RW	NA	System time Residue value (defined in $2^{(-32)}$ ns resolution).

8.2.2.19.4 Increment attributes register - TIMEINCA (0x00008C14)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
INCVALUE	30:0	0x0	RW	NA	Increment value to the SYSTIME registers on each 12.5 nanosecond
ISGN	31	0b	RW	NA	Increment sign. 0 - Each 12.5 ns cycle add to SYSTIM a value of 12.5 ns + Incvalue * $2^{(-32)}$ ns. 1 - Each 12.5 ns cycle add to SYSTIM a value of 12.5 ns - Incvalue * $2^{(-32)}$ ns.

8.2.2.19.5 Time adjustment offset register - TIMADJ (0x00008C18)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TADJL	30:0	0x00	RW	NA	Time adjustment value (defined in nsec units).
SIGN	31	0b	RW	NA	Sign (0b = "+", 1b = "-")

8.2.2.19.6 Time Sync Interrupt Mask Register - TSIM (0x00008C68)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
SYS_WARP	0	0b	RW	NA	SYSTIM Warp around Mask 0: No Interrupt generated when TSICR.SWARP is set. 1: Interrupt generated when TSICR.SWARP is set.
TXTS	1	0b	RW	NA	Transmit Time Stamp Mask 0: No Interrupt generated when TSICR.TXTS is set. 1: Interrupt generated when TSICR.TXTS is set.
RXTS	2	0b	RW	NA	Receive Time Stamp Mask 0: No Interrupt generated when TSICR.RXTS is set. 1: Interrupt generated when TSICR.RXTS is set.
TT0	3	0b	RW	NA	Target time 0 Trigger Mask 0: No Interrupt generated when TSICR.TT0 is set. 1: Interrupt generated when TSICR.TT0 is set.
TT1	4	0b	RW	NA	Target time 1 Trigger Mask 0: No Interrupt generated when TSICR.TT1 is set. 1: Interrupt generated when TSICR.TT1 is set.
AUTT0	5	0b	RW	NA	Auxiliary timestamp 0 taken Mask 0: No Interrupt generated when TSICR.AUTT0 is set. 1: Interrupt generated when TSICR.AUTT0 is set.
AUTT1	6	0b	RW	NA	Auxiliary timestamp 1 taken Mask 0: No Interrupt generated when TSICR.AUTT1 is set. 1: Interrupt generated when TSICR.AUTT1 is set.
TADJ	7	0b	RW	NA	Time Adjust 0 done mask 0: No Interrupt generated when TSICR.TADJ is set. 1: Interrupt generated when TSICR.TADJ is set.
RESERVED	31:8	0x0	RSV	NA	Reserved. Write 0, ignore on read.

8.2.2.19.7 Time Sync Interrupt Cause Register - TSICR (0x00008C60)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
SYS_WARP	0	0b	RW1C	DYNAMIC	SYSTIMEL Warp around. Set when SYSTIME ns counter warps around (SYSTIMEL). Warp around occurrence can be used by Software to update software time. This event should happen every second.
TXTS	1	0b	RW1C	DYNAMIC	Transmit Time Stamp. Set when new timestamp is loaded into TXSTMP register
RXTS	2	0b	RW1C	DYNAMIC	Receive Time Stamp. Set when new timestamp is loaded into RXSTMP register
TT0	3	0b	RW1C	DYNAMIC	Target time 0 trigger. Set when Target Time 0 (TRGTTIML/H0) trigger occurs.
TT1	4	0b	RW1C	DYNAMIC	Target time 1 trigger. Set when Target Time 1 (TRGTTIML/H1) trigger occurs.

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
AUTT0	5	0b	RW1C	DYNAMIC	Auxiliary timestamp 0 taken. Set when new timestamp is loaded into AUXSTMP 0 (auxiliary timestamp 0) register.
AUTT1	6	0b	RW1C	DYNAMIC	Auxiliary timestamp 1 taken. Set when new timestamp is loaded into AUXSTMP 1 (auxiliary timestamp 1) register.
TADJ	7	0b	RW1C	DYNAMIC	Time Adjust 0 done: Set when Time Adjust to clock out 0 or 1 completed
RESERVED	31:8	0x0	RSV	NA	Reserved. Write 0, ignore on read.

8.2.2.19.8 Tx Time Sync Control register - TSYNCTXCTL (0x00008C00)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TXTT	0	0b	ROS	NA	Tx timestamp valid (= 1b when a valid value for Tx timestamp is captured in the Tx timestamp register; clear by read of Tx timestamp register TXSTMPH).
RESERVED	3:1	0x0	RSV	NA	Reserved.
EN	4	0b	RW	NA	Enable Tx Timestamp. 0x0 = time stamping disabled. 0x1 = time stamping enabled.
RESERVED	5				Reserved
RESERVED	6				Reserved
RESERVED	7				Reserved
RESERVED	31:8	0x0	RSV	NA	Reserved.

8.2.2.19.9 Tx Timestamp Value Low - TXSTMPL (0x00008C04)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TXSTMPL	29:0	0x0	RO	NA	Tx Timestamp LSB Value (Defined in ns Units). The value in this field wraps around at 999999999 decimal.
RESERVED	31:30	0x0	RO	NA	Reserved.

8.2.2.19.10 Tx Timestamp Value High - TXSTMPH (0x00008C08)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TXSTMPH	31:0	0x0	RO	NA	Tx timestamp MSB value defined in seconds.

8.2.2.19.11 Rx Time Sync Control register - TSYNCRXCTL (0x00005188)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RXTT	0	0b	RO	NA	Rx timestamp valid (= 1b when a valid value for Rx timestamp is captured in the Rx timestamp register; clear by read of Rx timestamp register RXSTMPH).
TYPE	3:1	0x0	RW	NA	Type of packets to timestamp: 000b = Time stamp L2 (V2) packets only (Sync or Delay_req depends on RXMTRL.MSGT and packets with message ID 2 and 3). 001b = Time stamp L4 (V1) packets only (Sync or Delay_req depends on RXMTRL.CTRLT). 010b = Time stamp V2 (L2 and L4) packets (Sync or Delay_req depends on RXMTRL.MSGT and packets with message ID 2 and 3). 100b = Time stamp all packets. 101b - Time stamp V2 packets in which message ID bit 3 is zero, which means time stamp all event packets. 011b, 110b and 111b = Reserved.
EN	4	0b	RW	NA	Enable Rx Timestamp in TXSTMPH/L registers. 0x0 = Time stamping to TXSTMPH/L disabled. 0x1 = Time stamping to TXSTMPH/L enabled.
RESERVED	22:5	0x0	RSV	NA	Reserved.
TSIP_UT_EN	23	0b	RW	NA	Defines if untagged packets are appended a timestamp.
TSIP_UP_EN	31:24	0x0	RW	NA	Defines which UP timestamp is appended to the received packet (per UP bitmap). For example, to require a timestamp on packets received with UP = 0 or UP = 3, bits 24 and 26 should be set.

8.2.2.19.12 Rx timestamp Low - RXSTMPL (0x000051E8)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RXSTMPL	29:0	0x0	RO	NA	Rx timestamp LSB value (defined in nsec units). The value in this field wraps around at 999999999 decimal
RESERVED	31:30	0x0	RO	NA	Reserved

8.2.2.19.13 Rx timestamp High - RXSTMPH (0x000051A4)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RXSTMPH	31:0	0x0	RO	NA	Rx timestamp MSB value. defined in seconds

8.2.2.19.14 Rx message type register low - RXMTRL (0x00005120)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
CTRLT	7:0	0x0	RW	NA	V1 control to timestamp
MSGT	15:8	0x0	RW	NA	V2 messageId to timestamp
UDPT	31:16	0x13F	RW	NA	UDP port number to time stamp

8.2.2.19.15 Time Sync SDP Configuration Register - TSSDP (0x0000003C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
AUX0_SDP_SEL	1:0	0x0	RW	NA	Select one of the SDPs to serve as the trigger for auxiliary time stamp (AUXSTMPL0 and AUXSTMPH0 registers) 00b = SDP0 is assigned 01b = SDP1 is assigned 10b = SDP2 is assigned 11b = SDP3 is assigned
AUX0_TS_SDP_EN	2	0b	RW	NA	When set indicates that one of the SDPs can be used as an external trigger to Aux timestamp 0. Note: If this bit is set to one of the SDP pins, the corresponding pin should be configured to input mode using SDP_DIR.
AUX1_SDP_SEL	4:3	0x0	RW	NA	Select one of the SDPs to serve as the trigger for auxiliary time stamp 1 (in AUXSTMPL1 and AUXSTMPH1 registers) 00b = SDP0 is assigned 01b = SDP1 is assigned 10b = SDP2 is assigned 11b = SDP3 is assigned
AUX1_TS_SDP_EN	5	0b	RW	NA	When set indicates that one of the SDPs can be used as an external trigger to Aux timestamp 1. Note: If this bit is set to one of the SDP pins, the corresponding pin should be configured to input mode using SDP_DIR.
TS_SDP0_SEL	7:6	0x0	RW	NA	SDP0 allocation to Tsync event: When TS_SDP0_EN is set, these bits select the Tsync event that is routed to SDP0: 00b = Target Time 0 is output on SDP0 01b = Target Time 1 is output on SDP0 10b = Freq Clock 0 is output on SDP0 11b = Freq Clock 1 is output on SDP0
TS_SDP0_EN	8	0b	RW	NA	When set indicates that SDP0 is assigned to Tsync.
TS_SDP1_SEL	10:9	0x0	RW	NA	SDP1 allocation to Tsync event: When TS_SDP1_EN is set, these bits select the Tsync event that is routed to SDP1. 00b = Target Time 0 is output on SDP1 01b = Target Time 1 is output on SDP1 10b = Freq Clock 0 is output on SDP1 11b = Freq Clock 1 is output on SDP1
TS_SDP1_EN	11	0b	RW	NA	When set indicates that SDP1 is assigned to Tsync.

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TS_SDP2_SE L	13:12	0x0	RW	NA	SDP2 allocation to Tsync event: When TS_SDP2_EN is set, these bits select the Tsync event that is routed to SDP2. 00b = Target Time 0 is output on SDP2 01b = Target Time 1 is output on SDP2 10b = Freq Clock 0 is output on SDP2 11b = Freq Clock 1 is output on SDP2
TS_SDP2_EN	14	0b	RW	NA	When set indicates that SDP2 is assigned to Tsync.
TS_SDP3_SE L	16:15	0x0	RW	NA	SDP3 allocation to Tsync event: When TS_SDP3_EN is set, these bits select the Tsync event that is routed to SDP3. 00b = Target Time 0 is output on SDP3 01b = Target Time 1 is output on SDP3 10b = Freq Clock 0 is output on SDP3 11b = Freq Clock 1 is output on SDP3
TS_SDP3_EN	17	0b	RW	NA	When set indicates that SDP3 is assigned to Tsync.
RESERVED	31:18	0x0	RSV	NA	Reserved. Write 0, ignore on read.

8.2.2.19.16 TimeSync Auxiliary control register - TSAUXC (0x00008C20)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
EN_TT0	0	0b	RW	NA	Enable target time 0. Enable bit is set by software to 1b, to enable pulse or level change generation as a function of the TSAUXC.PLSG bit. Note: Target Time 0 is mutually exclusive with Clock out. Therefore, setting both EN_CLK and EN_TT0 is forbidden.
EN_TT1	1	0b	RW	NA	Enable target time 1: Enable bit is set by software to 1b, to enable pulse or level change generation as a function of the TSAUXC.PLSG bit.
EN_CLK0	2	0b	RW	NA	Enable Configurable Frequency Clock 0 Clock is generated according to frequency defined in the FREQOUT0 register on the SDP pin (0 to 3) that has both: 1. TSSDP.TS_SDPx_SEL field with a value of 10b. 2. TSSDP.TS_SDPx_EN value of 1b.
SAMP_AUTO	3	0b	SC	NA	When setting the SAMP_AUTO flag the SYSTIMEL/H registers are latched to the AUXSTMPL0/AUXSTMPH0 registers. Then the SAMP_AUTO flag is auto-cleared by the hardware.
ST0	4	0b	RW	NA	Start Clock 0 Toggle on Target Time 0 Enable Clock 0 toggle only after target time 0, that's defined in the TRGTTIML0 and TRGTTIMH0 registers, has passed.
EN_CLK1	5	0b	RW	NA	Enable Configurable Frequency Clock 1 Clock is generated according to frequency defined in the FREQOUT1 register on the SDP pin (0 to 3) that has both: 1. TSSDP.TS_SDPx_SEL field with a value of 11b. 2. TSSDP.TS_SDPx_EN value of 1b.
SAMP_AUT1	6	0b	SC	NA	When setting the SAMP_AUT1 flag the SYSTIMEL/H registers are latched to the AUXSTMPL1/AUXSTMPH1 registers. Then the SAMP_AUT1 flag is auto-cleared by the hardware.

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
ST1	7	0b	RW	NA	Start Clock 1 Toggle on Target Time 1 Enable Clock 1 toggle only after Target Time 1, that's defined in the TRGTTIML1 and TRGTTIMH1 registers, has passed.
EN_TS0	8	0b	RW	NA	Enable hardware time stamp 0 Enable Timestamping occurrence of change in SDP pin into the AUXSTMPL0 and AUXSTMPH0 registers. SDP pin (0 to 3) is selected for time stamping, if the SDP pin is selected via the TSSDP.AUX0_SDP_SEL field and the TSSDP.AUX0_TS_SDP_EN bit is set to 1b.
AUTT0	9	0b	RW	NA	Auxiliary timestamp taken. This is a read only bit field. At 0b the Auxiliary Time Stamp 0 is enabled. This bit is set to 1b by hardware following a level change of the input SDP. Reading the AUXSTMPH0 register clears this bit.
EN_TS1	10	0b	RW	NA	Enable hardware time stamp 1
AUTT1	11	0b	RW	NA	Auxiliary timestamp taken. This is a read only bit field. At 0b the Auxiliary Time Stamp 1 is enabled. This bit is set to 1b by hardware following a level change of the input SDP. Reading the AUXSTMPH1 register clears this bit.
RESERVED	16:12	0x0	RSV	NA	Reserved.
PLSG0	17	0b	RW	NA	Use Target Time 0 to generate start of pulse and Target Time 1 to generate end of pulse. SDP pin selected to drive pulse or level change is set according to the TSSDP.TS_SDPx_SEL field with a value of 00b and TSSDP.TS_SDPx_EN bit with a value of 1b. 0 = Target Time 0 generates change in SDP level. 1 = Target time 0 generates start of pulse on SDP pin. Note: Pulse or level change is generated when TSAUXC.EN_TT0 is set to 1.
RESERVED	30:18	0x0	RSV	NA	Reserved.
DISABLE_SYSTIME	31	1b	RW	NA	Disable SYSTIME count operation 0b - SYSTIME timer activated 1b - SYSTIME timer disabled. Value of SYSTIMEH, SYSTIMEL and SYSTIMER remains constant.

8.2.2.19.17 Target time register 0 low - TRGTTIMELO (0x00008C24)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TTL	29:0	0x0	RW	NA	Target Time 0 LSB register (defined in nsec units)
RESERVED	31:30	0x0	RW	NA	Reserved

8.2.2.19.18 Target time register 0 high - TRGTTIMEHO (0x00008C28)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TTH	31:0	0x0	RW	NA	Target Time 0 MSB register (defined in sec units)

8.2.2.19.19 Target time register 1 low - TRGTTIMEL1 (0x00008C2C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TTL	29:0	0x0	RW	NA	Target Time 1 LSB register (defined in nsec units)
RESERVED	31:30	0x0	RW	NA	Reserved

8.2.2.19.20 Target time register 1 high - TRGTTIMEH1 (0x00008C30)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TTH	31:0	0x0	RW	NA	Target Time 1 MSB register (defined in sec units)

8.2.2.19.21 Frequency Out 0 Control Register - FREQOUT0 (0x00008C34)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
CHCT	29:0	0x0	RW	NA	Clock Out Half Cycle Time. Defines the Half Cycle time of Clock-0 in ns units when clock output is enabled. Values permitted are between 6.4ns to 1sec. Note: Due to the 3.2ns cycle time accuracy of the internal clock the configured cycle time will be the average cycle time.
RESERVED	31:30	0x0	RSV	NA	Reserved

8.2.2.19.22 Frequency Out 1 Control Register - FREQOUT1 (0x00008C38)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
CHCT	29:0	0x0	RW	NA	Clock Out Half Cycle Time. Defines the Half Cycle time of Clock-1 in ns units when clock output is enabled. Values permitted are between 6.4ns to 1sec. Note: Due to the 3.2ns cycle time accuracy of the internal clock the configured cycle time will be the average cycle time.
RESERVED	31:30	0x0	RSV	NA	Reserved.

8.2.2.19.23 Auxiliary time stamp 0 register low - AUXSTMPL0 (0x00008C3C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TST_LOW	29:0	0x0	RO	NA	Auxiliary Time Stamp 0 LSB value (defined in nsec units)
RESERVED	31:30	0x0	RO	NA	Reserved

8.2.2.19.24 Auxiliary time stamp 0 register high - AUXSTMPH0 (0x00008C40)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TST_HI	31:0	0x0	RO	NA	Auxiliary Time Stamp 0 MSB value (defined in sec units)

8.2.2.19.25 Auxiliary time stamp 1 register low - AUXSTMPL1 (0x00008C44)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TST_LOW	29:0	0x0	RO	NA	Auxiliary Time Stamp 1 LSB value (defined in nsec units)
RESERVED	31:30	0x0	RO	NA	Reserved

8.2.2.19.26 Auxiliary time stamp 1 register high - AUXSTMPH1 (0x00008C48)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TST_HI	31:0	0x0	RO	NA	Auxiliary Time Stamp 1 MSB value (defined in sec units)

8.2.2.20 Virtualization PF Registers

8.2.2.20.1 Filter Local Packets Low - PFFLPL (0x000050B0)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FLP	31:0	0x0	RW	NA	Filter local packets - filter incoming packets whose MAC source address matches one of the LAN port DA MAC addresses. If the SA of the received packet matches one of the DA in the RAH/RAL registers, then the VM tied to this DA does not receive the packet. Other VMs can still receive it. (Bit per Pool)

8.2.2.20.2 Filter Local Packets High - PFFLPH (0x000050B4)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FLP	31:0	0x0	RW	NA	Filter local packets - filter incoming packets whose MAC source address matches one of the LAN port DA MAC addresses. If the SA of the received packet matches one of the DA in the RAH/RAL registers, then the VM tied to this DA does not receive the packet. Other VMs can still receive it. (Bit per Pool)

8.2.2.20.3 PF Virtual Control register - PFVTCTL (0x000051B0)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VT_ENA	0	0b	RW	NA	Virtualization Enabled Mode. When set, the PF supports either 16, 32, or 64 Pools. When cleared, Rx traffic is handled internally as if it belongs to VF zero while VF zero is enabled. This bit should be set the same as MTQC.VT_Ena.
RESERVED	6:1	0x0	RSV	NA	Reserved.
DEF_PL	12:7	0x0	RW	NA	Default Pool - Pool assignment for packets that do not pass any pool queuing decision. Enabled by the Dis_Def_Pool bit.
RESERVED	15:13	0x0	RSV	NA	Reserved.
POOLING_MODE	17:16	0x0	RW	NA	Pooling Mode 00b: Pool select by MAC address 01b: Pool select by MAC address or E-tag 10b: Pool select by MAC address or VNTag 11b: Reserved
RESERVED	28:18	0x0	RSV	NA	Reserved.
DIS_DEF_POOL	29	0b	RW	NA	Disable Default Pool - Determines the behavior of an Rx packet that does not match any Rx filter and is therefore not allocated a destination pool. 0b - Packet is assigned to the Default Pool (see DEF_PL above). 1b - Packet is dropped.

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RPL_EN	30	0b	RW	NA	Replication Enable when set to 1b. When set, MRQC.MRQE should be set to one of the virtualization modes (1000b - 1111b)
RESERVED	31	0b	RSV	NA	Reserved.

8.2.2.20.4 PF Mailbox - PFMAILBOX[n] (0x00004B00 + 0x4*n, n=0...63)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
STS	0	0b	WO	NA	Status/Command from PF ready. Setting this bit, causes an interrupt to the relevant VF. This bit always read as zero. Setting this bit will set the PFSTS bit in VFMailbox.
ACK	1	0b	WO	NA	VF message received. Setting this bit, causes an interrupt to the relevant VF... This bit always read as zero. Setting this bit will set the PFACK bit in VFMailbox.
VFU	2	0b	RW	NA	Buffer is taken by VF. This bit is RO for the PF and is a mirror of the VFU bit of the VFMailbox register.
PFU	3	0b	RW	NA	Buffer is taken by PF. This bit can be set only if the VFU bit is cleared and is mirrored in the PFU bit of the VFMailbox register.
RVFU	4	0b	WO	NA	Reset VFU - setting this bit will clear the VFU bit in the corresponding VFMailbox register - this bit should be used only if the VF driver is stuck. Setting this bit will also reset the corresponding bits in the PFMBICR, VFREQ, and VFACK fields.
RESERVED	31:5	0x0	RSV	NA	Reserved.

8.2.2.20.5 PF Mailbox Interrupt Causes Register - PFMBICR[n] (0x00000710 + 0x4*n, n=0...3)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VFREQ	15:0	0x0	RW1C	DYNAMIC	Each bit in the VFREQ field is set when VF number (16*n+j) wrote a message in its mailbox. While 'n' is the register index, n=0...3 and 'j' is the index of the bits in the VFREQ, j=0...15.
VFACK	31:16	0x0	RW1C	DYNAMIC	Each bit in the VFACK field is set when VF number (16*n+j) acknowledged a PF message. While 'n' is the register index, n=0...3 and '16+j' is the index of the bits in the VFACK, j=0...15.

8.2.2.20.6 PF Mailbox Interrupt Mask Register - PFMBIMR[n] (0x00000720 + 0x4*n, n=0...1)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VFIM	31:0	0xFFFFFFFF	RW	NA	Bit j - Mailbox indication from VF # (32*n+j) might cause an interrupt to the PF...

8.2.2.20.7 PF VFLR Events Clear - PFVFLREC[n] (0x00000700 + 0x4*n, n=0...1)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
CLEAR_VFLE	31:0	0x0	RW1C	DYNAMIC	When set, bit 'i' in register 'n' reflects an FLR event on VF# 32*n+i. These bits are accessible only to the PF and are cleared by writing 1.

8.2.2.20.8 PF VF Receive Enable - PFVFRE[n] (0x000051E0 + 0x4*n, n=0...1)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PFVFRE	31:0	0x0	RW	NA	Bit j - Enables receiving packets to VF# (32*n+j). Each bit is cleared by the relevant VFLR.

8.2.2.20.9 PF VF Transmit Enable - PFVFTE[n] (0x00008110 + 0x4*n, n=0...1)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PFVFTE	31:0	0x0	RW	NA	Bit j - Enables transmitting packets from VF# (32*n+j). Each bit is cleared by the relevant VFLR.

8.2.2.20.10 PF VM 0:31 Error Count Mask - PFVMECM0 (0x00008790)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FILTER	31:0	0x0	RW	NA	Defines if a packet dropped from pools 0 to 31 respectively is counted in the SSVPC counter.

8.2.2.20.11 PF VM 32:63 Error Count Mask - PFVMECM1 (0x00008794)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FILTER	31:0	0x0	RW	NA	Defines if a packet dropped from pools 32 to 63 respectively is counted in the SSVPC counter.

8.2.2.20.12 PF Queue Drop Enable Register - PFQDE (0x00002F04)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PFQDE	0	0b	RW	NA	Enable drop of packets from Rx queue Queue_Index. This bit overrides the SRRCTL.drop_en bit of each queue. I.e. if either of the bits is set, a packet received when no descriptor is available is dropped.
HIDE_VLAN	1	0b	RW	NA	If this bit is set, the RXDCTL.VME setting is ignored, the VLAN is always stripped, a value of zero is written in the RDESC.VLAN tag and in the RDESC.STATUS.VP fields of the received descriptor
STRIP_TAG	2	0b	RW	NA	If this bit is set, the E-tag is stripped from the packet
RESERVED	7:3	0x0	RSV	NA	Reserved.
QUEUE_INDEX	14:8	0x0	RW	NA	Indicates the queue referenced upon WE/RE commands.
RESERVED	15	0b	RSV	NA	Reserved.
WE	16	0b	RW	NA	Write Enable - When this bit is set, the content of bits 2:0 are written into the relevant queue context. Bit 3 is reserved. This bit should never be set together with the RE bit in this register.
RE	17	0b	RW	NA	Read Enable - When this bit is set, the content of bits 2:0 are read from the relevant queue context. Bits 3 is reserved. This bit should never be set together with the WE bit in this register.
RESERVED	31:18	0x0	RSV	NA	Reserved.

8.2.2.20.13 PF VM Tx Switch Loopback Enable - PFVMTXSW[n] (0x00005180 + 0x4*n, n=0...1)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
LLE	31:0	0x0	RW	NA	Local loopback enable. For each register 'n', and bit 'i', i=0..31, enables Local loopback for pool 32*n+1. When set, a packet originating from a specific pool and destined to the same pool is allowed to be looped back. If cleared, the packet is dropped.

8.2.2.20.14 PFVF Anti Spoof Control - PFVFSPOOF[n] (0x00008200 + 0x4*n, n=0...7)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MACAS	7:0	0x00	RW	NA	For each register 'n', and bit 'i', i=0..7, Enables anti-spoofing filter on Ethernet MAC Addresses for VF(8*n+i).
VLANAS	15:8	0x00	RW	NA	For each register 'n', and bit '8+i', i=0..7, Enables anti-spoofing filter on VLAN tag for VF(8*n+i). Note: If VLANAS is set for a specific pool, then the respective MACAS bit must be set as well.
ETHERTYPEAS	23:16	0x0	RW	NA	For each register 'n', and bit '16+i', i=0..7, Enables anti-spoofing filter on EtherType for VF(8*n+i).
ETHERTYPELBS	31:24	0x0	RW	NA	For each register 'n', and bit '24+i', i=0..7, Enables Loop-Back filter on EtherType for VF(8*n+i).

8.2.2.20.15 PF DMA Tx General Switch Control - PFDTXGSWC (0x00008220)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
LBE	0	0b	RW	NA	Enables VMDQ Loopback.
RESERVED	31:1	0x0	RSV	NA	Reserved.

8.2.2.20.16 PF VM Tag Insert Register - PFVMTIR[n] (0x00017000 + 0x4*n, n=0...63)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PORT_TAG_ID	31:0	0x0	RW	NA	Port tag to insert if PFVMTIR.TAGA field = 01b or 10b. TAGA = 01b - insert E-tag i.e. EtagEtype-PFVMTIR[31:0]-0x0000 TAGA = 10b - insert Vntag i.e. VntagEtype-PFVMTIR[31:0] TAGA = Other - ignore this field.

8.2.2.20.17 PF VM VLAN Insert Register - PFVMVIR[n] (0x00008000 + 0x4*n, n=0...63)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PORT_VLAN_ID	15:0	0x0	RW	NA	Port VLAN tag to insert if the VLANA field = 01b.
RESERVED	26:16	0x0	RSV	NA	Reserved.
TAGA	28:27	0x0	RW	NA	Tag Action 00b = Do not insert any outer tag 01b = Insert an E-tag 10b = Insert a VN-tag 11b = Reserved
RESERVED	29	0b	RSV	NA	
VLANA	31:30	0x0	RW	NA	VLAN action: 00 - use descriptor command 01 - always insert Default VLAN 10 - Never insert VLAN 11 - Reserved.

8.2.2.20.18 PF VM L2Control Register - PFVML2FLT[n] (0x0000F000 + 0x4*n, n=0...63)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	21:0	0x00	RSV	NA	Reserved.
UPE	22	0b	RW	NA	Unicast Promiscuous
VPE	23	0b	RW	NA	VLAN Promiscuous Enable
AUPE	24	0b	RW	NA	Accept Untagged packets enable: When set, packets without VLAN tag can be forwarded to this queue, assuming they pass the Ethernet MAC Address queuing mechanism.
ROMPE	25	0b	RW	NA	Receive overflow multicast packets: Accept packets that match the MTA table.
ROPE	26	0b	RW	NA	Receive MAC filters overflow: Accept packets that match the PFUTA table.
BAM	27	0b	RW	NA	Broadcast accept
MPE	28	0b	RW	NA	Multicast promiscuous
RESERVED	31:29	0x00	RSV	NA	Reserved.

8.2.2.20.19 PF VM VLAN Pool Filter - PFVLVF[n] (0x0000F100 + 0x4*n, n=0...63)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VLAN_ID	11:0	X	RW	NA	Defines a VLAN tag for pool VLAN filter n. The bitmap defines which pools belong to this VLAN. Note: Appears in Little Endian order (LS byte last on the wire).
RESERVED	30:12	X	RSV	NA	Reserved.
VI_EN	31	0b	RW	NA	VLAN Id Enable -this filter is valid.

8.2.2.20.20 PF VM VLAN Pool Filter Bitmap - PFVLVFB[n] (0x0000F200 + 0x4*n, n=0...127)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
POOL_ENA	31:0	X	RW	NA	Pool Enable bit array. Each couple of registers '2*n' and '2*n+1' enables routing of packets that match a PFVLVF[n] filter to a Pool list. Each bit when set, enables packet reception with the associated Pools as described below: Bit 'i' in register '2*n' is associated with POOL 'iTab' Bit 'i' in register '2*n+1' is associated with POOL '32+i'.

8.2.2.20.21 PF Unicast Table Array - PFUTA[n] (0x0000F400 + 0x4*n, n=0...127)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
BIT_VECTOR	31:0	X	RW	NA	Word wide bit vector specifying 32 bits in the unicast destination address filter table.

8.2.2.20.22 PF Mirror Rule Control - PFMRCTL[n] (0x0000F600 + 0x4*n, n=0...3)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VPME	0	0b	RW	NA	Virtual Pool mirroring enable -enables mirroring of certain pools as defined in the PFMRVM registers.
UPME	1	0b	RW	NA	Uplink port mirroring enable -enables mirroring of all traffic received from the network.
DPME	2	0b	RW	NA	Downlink port mirroring enable: Enables mirroring of all traffic transmitted to the network.
VLME	3	0b	RW	NA	VLAN mirroring enable -enables mirroring of a set of given VLANs as defined in the PFMRVLAN registers.
RESERVED	7:4	0x0	RSV	NA	Reserved.
MP	13:8	0x0	RW	NA	Mirror Pool: Defines the destination pool for this mirror rule.
RESERVED	31:14	0x0	RSV	NA	Reserved.

8.2.2.20.23 PF Mirror Rule VLAN - PFMRVLAN[n] (0x0000F610 + 0x4*n, n=0...7)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VLAN	31:0	0x0	RW	NA	Bitmap listing which VLANs participate in the mirror rule.

8.2.2.20.24 PF Mirror Rule Pool - PFMRVM[n] (0x0000F630 + 0x4*n, n=0...7)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
POOL	31:0	0x0	RW	NA	Bitmap listing which pools participate in the mirror rule.

8.2.2.20.25 Last VM Misbehavior Cause - TX - LVMMC_TX (0x00008108)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MAC_HEADER	0	0b	RC	NA	Illegal MAC header size.
IPV4_HEADE R	1	0b	RC	NA	Illegal IPv4 header size.
IPV6_HEADE R	2	0b	RC	NA	Illegal IPv6 header size.
WRONG_MAC _IP	3	0b	RC	NA	Wrong MAC +IP header size
TCP_LSO	4	0b	RC	NA	Illegal TCP header was detected in a large send operation.
RESERVED	5				Reserved
UDP_LSO	6	0b	RC	NA	Illegal UDP header was detected in a large send operation.
STCP_CS	7	0b	RC	NA	Illegal STCP header was detected in an SCTP checksum offload operation,
SIZE	8	0b	RC	NA	Illegal packet size (> 15.5K).
RESERVED	9	0b	RSV	NA	Reserved
OFF_ILL	10	0b	RC	NA	Illegal offload request.
SCTP_ALIGN ED	11	0b	RC	NA	SCTP CRC request of non 4 byte aligned data.
ZERO_MSS	12	0b	RC	NA	A request for large send with zero MSS was detected.
CONTEXT_IN _PACKET	13	0b	RC	NA	A context descriptor was detected in the middle of a packet.
LSO_MORE_T HAN_4	14	0b	RC	NA	Large send with more than 4 header buffers misbehavior was detected.
OOS_SSO	15	0b	RC	NA	Out of sync during Single Send.
OOS_LSO	16	0b	RC	NA	Out of sync during Large Send.
SSO_UDP	17	0b	RC	NA	Wrong parameter of headers for UDP SSO
SSO_TCP	18	0b	RC	NA	Wrong parameter of headers for TCP SSO
INV_MACC	19	0b	RC	NA	Invalid memory access A PCIe DMA access initiated by a VF ended with Unsupported Request (UR) or Completer Abort (CA) or was prevented from being sent if out of range. When a Malicious DMA access is detected the Tx queue (VF) that initiated the access is disabled and corresponding WQBR_TX bit is set.
DESC_TYPE	20	0b	RC	NA	Wrong descriptor type (other than 2,3)
WRONG_NUL L	21	0b	RC	NA	Null without EOP
NO_EOP	22	0b	RC	NA	Packet without EOP (i.e. bigger than the ring size)
CONTEXT_BU RST	23	0b	RC	NA	Contiguous context descriptor burst that exceeds 3 Contexts.
RESERVED	24	0b	RSV	NA	Reserved
MAC_VLAN_S POOF	25	0b	RC	NA	A MAC spoof or VLAN spoof attempt was detected
VLAN_IERR	26	0b	RC	NA	VLAN Insertion Error. VLE bit set in TX descriptor when VMVIR[n].VLANA register field is not 0 causing drop of TX packets.
LEGACY_DES C_IOV	27	0b	RC	NA	A legacy desc in IOV was detected

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	28				Reserved
INVALID_TXQ_CONTEXT	29	0b	RC	NA	An invalid transmit queue context was detected when queue was enabled or an attempt to change the static part of the queue context on the fly was detected.
RESERVED	31:30	0x0	RSV	NA	Reserved

Note: Only the first malicious event is registered for each packet, so if a bit is not set it doesn't mean that this event didn't occur, only that another malicious behavior was detected first.

8.2.2.20.26 Last VM Misbehavior Cause - RX - LVMMC_RX (0x00002FA8)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
INV_MACC	0	0b	RC	NA	Invalid memory access A PCIe DMA access initiated by a VF ended with Unsupported Request (UR) or Completer Abort (CA) or a read access was blocked due to out of range address. When a Malicious DMA access is detected the Rx queue (VF) that initiated the access is disabled and corresponding WQBR_RX bit is set.
INVALID_RX_Q_CONTEXT	1	0b	RC	NA	An invalid receive queue context was detected when queue was enabled or an attempt to change the static part of the queue context on the fly was detected.
RESERVED	31:2	0x0	RSV	NA	Reserved

8.2.2.20.27 Last Malicious VM - TX - LMVM_TX (0x00008124)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MALICIOUS_QUEUE	6:0	0x0	RW	NA	The Tx queue on which the malicious behavior reported in LVMMC was detected. The queue number is the absolute number in the PF space.
RESERVED	16:7	0x0	RSV	NA	
MAL_PF	17	0b	RW	NA	Malicious Driver behavior detected on current PF 0 = malicious event was on a queue belonging to a VF. 1 = malicious event was on a queue belonging to the PF.
RESERVED	31:18	0x0	RSV	NA	

8.2.2.20.28 Last Malicious VM - RX - LMVM_RX (0x00002FA4)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MALICIOUS_QUEUE	6:0	0x0	RW	NA	The Rx queue on which the malicious behavior reported in LVMMC was detected. The queue number is the absolute number in the PF space.
RESERVED	16:7	0x0	RSV	NA	
MAL_PF	17	0b	RW	NA	Malicious Driver behavior detected on current PF 0 = malicious event was on a queue belonging to a VF. 1 = malicious event was on a queue belonging to the PF.
RESERVED	31:18	0x0	RSV	NA	

8.2.2.20.29 Wrong Queue Behavior Register - Tx - WQBR_TX[n] (0x00008130 + 0x4*n, n=0...3)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
WVBR	31:0	0x0	RW1C	DYNAMIC	Bitmap indicating against which Tx queue an anti spoof action or malicious action was taken. The queue is released only by a reset of the VF or by clearing the corresponding bit in the WQBR_TX register

8.2.2.20.30 Wrong Queue Behavior Register - Rx - WQBR_RX[n] (0x00002FB0 + 0x4*n, n=0...3)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
WVBR	31:0	0x0	RW1C	DYNAMIC	Bitmap indicating against which Rx queue a malicious action was taken. The queue is released only by a reset of the VF or by clearing the corresponding bit in WQBR_RX register.

8.2.2.21 Power Management Registers

8.2.2.21.1 Latency Tolerance Reporting (LTR) Control - LTRC (0x00011708)

Reset source: PCIe In-band Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
SLTRV	9:0	0x5	RW	NA	Snoop Latency Value Along with the Max Snoop Latency Scale field, this register specifies the maximum snoop latency this function wants to request. Software should set this only if it is set to a lower number than the platform's maximum supported latency.
SSCALE	12:10	0x2	RW	NA	Snoop Latency Scale This field provides a scale for the value contained within the Snoop Latency Value field. Encoding: 000 = Value times 1ns 001 = Value times 32ns 010 = Value times 1,024ns 011 = Value times 32,768ns 100 = Value times 1,048,576ns 101 = Value times 33,554,432ns 110-111 = Not Permitted The SW driver is advised to use the same scale as configured in the LTR capability.
RESERVED	14:13	0x0	RSV	NA	Reserved Write 0, ignore on read.
LTRS_REQUIREMENT	15	0b	RW	NA	LTR Snoop requirement 0 - No Latency requirements in Snoop memory access. 1 - Latency tolerance in Snoop memory access specified in the LTRC.SLTRV field.
NSLTRV	25:16	0x5	RW	NA	No Snoop Latency Value Along with the Max No Snoop Latency Scale field, this register specifies the maximum no snoop latency this function wants to request. Software should set this only if it is set to a lower number than the platform's maximum supported latency.
NSSCALE	28:26	0x2	RW	NA	No Snoop Latency Scale This field provides a scale for the value contained within the No Snoop Latency Value field. Encoding: 000 = Value times 1ns 001 = Value times 32ns 010 = Value times 1,024ns 011 = Value times 32,768ns 100 = Value times 1,048,576ns 101 = Value times 33,554,432ns 110-111 = Not Permitted The SW driver is advised to use the same scale as configured in the LTR capability.
LTR_REQ	29	0b	RW	NA	0 - The NIC initiates an LTR message with "cleared" LTR requirements message 1 - The NIC initiates an LTR message with LTR requirements message

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
LTR_SEND	30	0b	RW	NA	Indication to send an LTR message This bit is set by the FW (indicates HW to send LTR message update to the system) and cleared by HW after the message was sent. This bit is not sent as part of the LTR message. the type of the message to be sent defined by LTR_REQ (bit #29)
LTRNS_REQUIREMENT	31	0b	RW	NA	LTR No Snoop requirement 0 - No Latency requirements in No Snoop memory access. 1 - Latency tolerance in No Snoop memory access specified in the LTRC.NSLTRV field.

8.2.2.21.2 Energy Efficient Ethernet (EEE) Register - EEER (0x000043A0)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TW_SYSTEM	15:0	0x0	RW	NA	Time expressed in microseconds that no data will be transmitted following move from EEE TX LPI link state to Link Active state. Field holds the Transmit Tw_sys_tx value negotiated during EEE LLDP negotiation. Note: - If value is lower than minimum Tw_sys_tx value defined in IEEE802.3az clause 78.5 then interval where no data is transmitted following move out of EEE TX LPI state defaults to minimum Tw_sys_tx. - Following link disconnect or Auto-negotiation value of this field returns to default value, until SW re-negotiates new tw_sys_tx value via EEE LLDP. - Fast Retrain and Local/Remote Fault indication are not considered link disconnect and do not cause the field to return to the default value. - When transmitting flow control frames device waits the minimum time defined in the IEEE802.3az standard before transmitting the flow control packet. device does not wait the Tw_system time following exit of LPI before transmitting the flow control frame.
TX_LPI_EN	16	0b	RW	NA	Enable entry into EEE LPI on TX path 0b - Disable entry into EEE LPI on TX path. 1b - Enable entry into EEE LPI on TX path. Notes: - Even when TX_LPI_EN is 1b device will not enable entry into TX LPI state for at least 1 second following the change of link_status to OK as defined in IEEE802.3az clause 78.1.2.1. - Even if the TX_LPI_EN bit is set, device will initiate entry into TX EEE LPI link state only if EEE support at the link speed was negotiated during Auto-negotiation.
RX_LPI_EN	17	1b	RW	NA	Enable entry into EEE LPI on RX path 0b - Disable entry into EEE LPI on RX path. 1b - Enable entry into EEE LPI on RX path. Note: Even if the RX_LPI_EN bit is set, device will recognize entry into RX EEE LPI link state only if EEE support at the link speed was negotiated during Auto-negotiation.

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	27:18	0x0	RSV	NA	Reserved
EEE_FRC_AN	28	0b	RW	NA	Force EEE Auto-negotiation When bit is set to 1 enables EEE operation in internal MAC logic even if link partner does not support EEE. Note: Should be set to 1b to enable testing of EEE operation via MAC loop-back
RESERVED	31:29	0x0	RSV	NA	Reserved Write 0 ignore on read.

8.2.2.21.3 Energy Efficient Ethernet (EEE) Setup Register - EEE_SU (0x00004380)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
DTW_MIN	7:0	0x0	RW	NA	Time to add to minimum Tw_sys_tx value defined in IEEE802.3az clause 78.5 This additional time should be configured according to the link speed/type and is in addition to the specification definition. For this product, the basic value of Tw_sys_tx is: 7.36 usec when link speed is 10G. An additional 10 usec should be added for worst case normal operation by setting this field to 0x64. 16.5 usec. when link speed is 1G. This value can be kept (keep this field at zero) for normal operation. The minimum Tw_sys_tx value defines the duration where no data will be transmitted following move out of EEE LPI TX state. Time to add defined in this field is expressed in 0.1 microsecond resolution. Note: The idle time value defined by this field plus the Tw_sys_tx value defined in IEEE802.3az clause 78.5 is used when moving out of EEE TX LPI state to transmit flow control frames even if value specified in EEER.Tw_system field is higher.
RESERVED	15:8	0x0	RSV	NA	Reserved
TW_WAKE_MIN	21:16	0x0	RW	NA	Minimum time expressed in 1 microseconds, between sending a request to move into EEE TX LPI and sending a request to move back to active state. Should be set to 0x7 for normal operation. Note: If conditions to exit LPI during the Tw_wake_min interval cease to exist then device will not move out of TX LPI after timer has expired.
RESERVED	23:22	0x0	RW	NA	Reserved
TX_LU_LPI_DELAY	25:24	0x3	RW	NA	Delay to enable entry of TX EEE LPI state following Link-up indication. 00b - No delay 01b - 10 ms 10b - 100 ms 11b - 1 Second Note: IEEE802.3az clause 78.1.2.1 defines delay of 1 second following link-up.

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
TEEE_DLY	31:26	0x2	RW	NA	TX EEE LPI Entry delay Field defines delay to EEE entry once conditions to enter EEE LPI are detected. Field resolution is 1 micro seconds. Note: If conditions to enter LPI during the TEEE_DLY interval cease to exist device will not enter TX LPI and continue normal operation.

8.2.2.21.4 Energy Efficient Ethernet (EEE) STATUS - EEE_STAT (0x00004398)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	28:0	0x0	RSV	NA	Reserved Write 0, ignore on read.
EEE_NEG	29	0b	RO	NA	EEE support Negotiated on link 0b - EEE operation not supported on link. 1b - EEE operation supported on link.
RX_LPI_STAT US	30	0b	RO	NA	RX Link in LPI state 0b - RX in Active state 1b - RX in LPI state
TX_LPI_STAT US	31	0b	RO	NA	TX Link in LPI state 0b - TX in Active state 1b - TX in LPI state

8.2.2.21.5 EEE TX LPI Count - TLPIC (0x000041F4)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
ETLPIC	31:0	0x0	RC	NA	Number of EEE TX LPI events. Register cleared on read, register doesn't wrap around at 0xFFFFFFFF.

8.2.2.21.6 EEE RX LPI Count - RLPIC (0x000041F8)

Reset source: Master Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
ERLPIC	31:0	0x0	RC	NA	Number of EEE RX LPI events. Register cleared on read, register doesn't wrap around at 0xFFFFFFFF.

8.2.2.22 Manageability (ARC Subsystem) HOST Interface Registers

Host interface to the ARC subsystem is described in the Manageability Host Interface Section in the Manageability chapter.

8.2.2.22.1 Software Semaphore Register - SWSM (0x00015F70)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
SMBI	0	0b	RWS	NA	Set on HOST Read, Can be cleared on HOST Write, Not Set Also cleared on PCI Immediate In-band Reset
RESERVED	31:1	0x0	RSV	NA	Reserved.

8.2.2.22.2 Firmware Semaphore Register - FWSM (0x00015F74)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	0				Reserved
FLASH_UPDATE_ENABLED	1	0b	RW	NA	If set, the Flash update functionality is active.
PASS_THROUGH_ENABLED	2	0b	RW	NA	If set, the pass through functionality is active.
RESERVED	3				Reserved
HOST_INTERFACE_ENABLED	4	0b	RW	NA	If set, the host interface functionality is active.
RESERVED	14:5	0x0	RSV	NA	Reserved.
FW_VAL_BIT	15	0b	RW	NA	Firmware Valid Bit. Hardware clears this bit in reset de-assertion so software can know firmware modes (bits 1-4) are invalid. Firmware should set it to 1b when it is ready (end of boot sequence).
RESERVED	18:16	0x0	RW	NA	Reserved

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
EXT_ERR_IND	24:19	0x0	RW	NA	External error indication firmware uses this register to store the reason that the firmware has reset / clock gated (such as NVM, patch corruption). Possible values: 0x00: No error. 0x01: Unspecified error. 0x02: No manageability (No NVM). 0x03: TCO isolate mode active. 0x05: Shadow RAM dump fault. 0x06: Bad Flash contents. (for FW/RO update) 0x10: NVM CRC error in "Test Configuration" module 0x11: NVM CRC error in "Common FW Parameters" module 0x12: NVM CRC error in "PT LAN 0 configuration" module 0x13: NVM CRC error in "Sideband configuration" module 0x14: NVM CRC error in "Flexible TCO filter configuration" module 0x15: NVM CRC error in "PT LAN 1 configuration" module 0x16: NVM CRC error in "OEM support structure" module 0x20: Management memory parity/ECC error. 0x21: SR ECC error 0x22-0x3E: Reserved 0x3F = Reserved (max error value).
RESERVED	25	0b	RSV	NA	Reserved
PHY0_CONFIG_ERR_IND	26	0b	RW	NA	PHY0 Configuration Error Indication. Set to 1b by firmware when it fails to configure PHY of LAN0. Cleared by firmware upon successful configuration of PHY of LAN0.
PHY1_CONFIG_ERR_IND	27	0b	RW	NA	PHY1 Configuration Error indication. Set to 1b by firmware when it fails to configure PHY of LAN1. Cleared by firmware upon successful configuration of PHY of LAN1
RESERVED	30:28	0x0	RSV	NA	Reserved.
FACTORY_MAC_ADDRESS_RESTORED	31	0b	RW	NA	When set, it indicates to software that the factory MAC address and the current MAC addresses are identical after the last power-up event. This bit is cleared by the device at power up.

Notes:

- This register should be written only by the manageability firmware. The device driver should only read this register.
- The firmware ignores the NVM semaphore in operating system hung states.
- Bits 15:0 are cleared on firmware reset.

8.2.2.22.3 Software-Firmware Synchronization - SW_FW_SYNC (0x00015F78)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
SW_NVM_SM	0	0b	RW	NA	If set, NVM access is owned by software
SW_PHY0_SM	1	0b	RW	NA	If set, PHY 0 access is owned by software
SW_PHY1_SM	2	0b	RW	NA	If set, PHY 1 access is owned by software
SW_MAC_CSR_SM	3	0b	RW	NA	If set, MAC CSR access is owned by software
RESERVED	4	0b	RSV	NA	Reserved
FW_NVM_SM	5	0b	RW	NA	If set, NVM access is owned by firmware
FW_PHY0_SM	6	0b	RW	NA	If set, PHY 0 access is owned by firmware
FW_PHY1_SM	7	0b	RW	NA	If set, PHY 1 access is owned by firmware
FW_MAC_CSR_SM	8	0b	RW	NA	If set, MAC CSR access is owned by firmware
NVM_UPDATE_STARTED	9	0b	RW	NA	If set, NVM update started, Software should not write to NVM.
SW_MNG_SM	10	0b	RW	NA	If set, Manageability host interface is owned by software.
RESERVED	30:11	0x0	RSV	NA	Reserved. for future use.
REGSMP	31	0b	RW	NA	Register Semaphore. This bit is used to semaphore the access to this register between the firmware and software with no hardware enforcement. When the bit value is 0b and the register is read, the returned value is zero and the bit setting reverts to 1b (for the next read cycle). Writing 0b to this bit clears it. A software driver that reads this register and gets the value of zero for this bit locks the access to this register until it clears this bit.

Notes:

- This register is shared by both LAN ports.
- See SW and FW synchronization Section for more details on software and firmware synchronization.

8.2.2.22.4 HOST Interface Control Register - HICR_HOST (0x00015F00)

Reset source: Management Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
EN	0	0b	RW	NA	Enable. When set it indicates that a RAM area is provided for device driver accesses. This bit is read only for the device driver.
C	1	0b	RW	NA	Command. The device driver sets this bit when it has finished putting a command block in the ARC internal data ram. This bit should be cleared by the firmware when the command's processing is completed. Setting this bit causes an interrupt to the ARC.
SV	2	0b	RW	NA	Status Valid. Indicates that there is a valid status in CSR area that the device driver can read. 0x1 -status valid. 0x0 -status not valid. The value of the bit is valid only when the C bit is cleared. Only the device driver reads this bit.
RESERVED	31:3	0x0	RSV	NA	Reserved.

8.2.2.22.5 Host ARC Data RAM - ARCRAM[n] (0x00015800 + 0x4*n, n=0...447)

Reset source: Management Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RAM_SPACE	31:0	X	RW	NA	RAM Space area that spans on the CSR space: 0x15800 - 0x15EFC

8.2.2.22.6 legacy Firmware Status register - FWSTS (0x00015F0C)

Reset source: Management Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	8:0	0x0	RSV	NA	Reserved.
FWRI	9	1b	RW	NA	Firmware Reset Indication. Set when firmware reset is asserted. Cleared when HOST writes '1' to it. Writing '0' to this bit does not change its value. Note: This bit is also set after LAN_PWR_GOOD. This bit is read-only through the AUX interface.
RESERVED	31:10	0x0	RSV	NA	Reserved.

8.2.2.23 Manageability Extension Registers

Host Interface registers.

8.2.2.23.1 Device Public Key - GL_MNG_ECDSA_PUBKEY[n] (0x00083300 + 0x4*n, n=0...11)

Reset source: Management Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GL_MNG_ECDSA_PUBKEY	31:0	0x0	RO	NA	SHA value calculated on signed image

8.2.2.23.2 Tools Mailbox HOST Interface Descriptor Area - GL_HIDA[n] (0x00082000 + 0x4*n, n=0...15)

Reset source: Management Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GL_HIDB	31:0	0x0	RW	NA	

8.2.2.23.3 Per-PF AQ Mailbox HOST Interface Descriptor Area - PF_HIDA[n] (0x00085000 + 0x4*n, n=0...15)

Reset source: Management Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GL_HIDB	31:0	0x0	RW	NA	

8.2.2.23.4 Measurement Status - GL_MNG_SHA_EXTEND_STATUS (0x00083148)

Reset source: Management Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
STAGE	2:0	0x0	RO	NA	FW load SHA reflected in registers: 000: No value 001: ROM version (in GL_MNG_SHA_EXTEND_ROM) 010: Mini loader SHA (in GL_MNG_SHA_EXTEND_ROM) 011: Mini loader SHA (no Full FW) (in GL_MNG_SHA_EXTEND_ROM) 100: Full FW SHA (in GL_MNG_SHA_EXTEND) and Mini loader SHA (in GL_MNG_SHA_EXTEND_ROM) 1XX: Reserved
RESERVED	29:3	0x0	RSV	NA	Reserved
FW_HALTED	30	0b	RO	NA	If set, firmware load process was halted before completion
DONE	31	0b	RO	NA	If set, indicates this is the final version of GL_MNG_SHA_EXTEND expected

8.2.2.23.5 Tools Mailbox HOST Interface Buffer Area - GL_HIBA[n] (0x00081000 + 0x4*n, n=0...1023)

Reset source: Management Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GL_HIBA	31:0	0x0	RW	NA	

8.2.2.23.6 NVM Bank Measurement - GL_MNG_SHA_EXTEND[n] (0x00083340 + 0x4*n, n=0...11)

Reset source: Management Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GL_MNG_SHA_EXTEND	31:0	0x0	RO	NA	SHA value calculated on signed image

8.2.2.23.7 Firmware Reset Count - GL_FWRESETCNT (0x00083100)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FWRESETCNT	31:0	0x0	RO	NA	Firmware resets count. Updated by Hardware. Saturates at 0xFFFF,FFFF

8.2.2.23.8 MNG FW RAM status registers - GL_MNG_FW_RAM_STAT (0x000830EC)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FW_RAM_RST_STAT	0	0b	RO	NA	FW RAM Reset State: Set by FW and cleared when RAM is initialized: 1. POR reset. 2. EMPR followed ECC reset in EMP memories.
MNG_MEM_ECC_ERR	1	0b	RO	NA	ECC error in one of MNG memories. Should be cleaned by FW.
RSV	31:2	0x0	RSV	NA	Reserved.

8.2.2.23.9 Per-PF AQ Mailbox HOST Interface Buffer Area - PF_HIBA[n] (0x00084000 + 0x4*n, n=0...1023)

Reset source: Management Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GL_HIBA	31:0	0x0	RW	NA	

8.2.2.23.10 Tools Mailbox HOST Interface Control Register - GL_HICR (0x00082040)

Reset source: Management Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	0	0b	RSV	NA	
C	1	0b	RW	NA	Command The tool sets this bit when it has finished putting a command block in the command buffer This bit should be cleared by the firmware when the command's processing is completed. Setting this bit causes an interrupt to the FW. This bit can be only set through the CSR (SW) interface and can be cleared through the AUX (FW) interface While this bit is set the memory is RO to SW.
SV	2	0b	RW	NA	Status Valid Indicates that there is a valid status in the 1st descriptor that the device driver can read. If Flags.BUF is set in the 1st descriptor, the buffer is valid 0b = Status not valid. 1b = Status valid.
EV	3	0b	RW	NA	Event Valid Indicates that there is a valid status in the 2nd descriptor that the device driver can read. If Flags.BUF is set in the 2nd descriptor, the buffer is valid 0b = Status not valid. 1b = Status valid.
RESERVED1	31:4	0x0	RSV	NA	

8.2.2.23.11 Mini-Loader Measurement - GL_MNG_SHA_EXTEND_ROM[n] (0x000832C0 + 0x4*n, n=0...11)

Reset source: Management Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GL_MNG_SHA_EXTEND_ROM	31:0	0x0	RO	NA	SHA value calculated on signed image

8.2.2.23.12 Tools Mailbox HOST Interface Enable Register - GL_HICR_EN (0x00082044)

Reset source: Management Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
EN	0	0b	RO	NA	Enable When set, indicates that the buffer is accessible for the device. Note: This bit is common to all nodes/functions.
RESERVED	31:1	0x0	RSV	NA	

8.2.2.23.13 AQ Mailbox HOST Interface Control Register - PF_HICR (0x00082048)

Reset source: Management Reset, NVM Load: No, Scope: PF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
EN	0	0b	RO	NA	Enable
C	1	0b	RW	NA	Command The tool sets this bit when it has finished putting a command block in the command buffer This bit should be cleared by the firmware when the command's processing is completed. Setting this bit causes an interrupt to the FW. This bit can be only set through the CSR (SW) interface and can be cleared through the AUX (FW) interface While this bit is set the memory is RO to SW.
SV	2	0b	RW	NA	Status Valid Indicates that there is a valid status in the 1st descriptor that the device driver can read. If Flags.BUF is set in the 1st descriptor, the buffer is valid 0b = Status not valid. 1b = Status valid.
EV	3	0b	RW	NA	Event Valid Indicates that there is a valid status in the 2nd descriptor that the device driver can read. If Flags.BUF is set in the 2nd descriptor, the buffer is valid 0b = Status not valid. 1b = Status valid.
EP	4	0b	RW	NA	Event Pending. At 1b one or more events are pending.
RESERVED	31:5	0x0	RSV	UNDEFINED	Reserved

8.2.2.23.14 Global DCB General Control - GLDCB_GENC (0x00083044)

Reset source: Master Reset

NVM Load: Type1/2 in Master Reset Register Autoload module, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PCIRTT	15:0	0x009C	RW	NA	PCIe Round Trip Time. It is expressed in 16 bytes units. Default is 2 us PCIe round trip time assuming 10G links (no optimization done for lower links speeds). For a 40G link, the number shall be 0x0270.
RESERVED	31:16	0x0	RSV	NA	

8.2.2.23.15 Port DCB General Control - PRTDCB_GENC (0x00083000)

Reset source: Master Reset, NVM Load: No, Scope: PRT

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	1:0	0x0	RSV	NA	Reserved
NUMTC	5:2	0x1	RW	NA	Number of Traffic Classes (TCs) for the port. This field shall be set consistently with the settings made in the Tx-scheduler.
RESERVED	15:6	0x0	RSV	NA	Reserved
PFCLDA	31:16	0x079D	RW	NA	PFC Link Delay Allowance. It is expressed in 16 bytes units. Default value assumes 9.5 KB Jumbo frames over a 10G link with a 10GBASE-T PHY and 100 meter Cat6 cable (no optimization done for lower links speeds). For a 40G link, the number shall be 0x1E70.

8.2.2.23.16 Port DCB General Status - PRTDCB_GENS (0x00083020)

Reset source: Master Reset, NVM Load: No, Scope: PRT

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
DCBX_STATUS	2:0	0x0	RW	NA	DCBX status. 0x0: NOT_STARTED 0x1: IN_PROGRESS 0x2: DONE 0x3: MULTIPLE_PEERS 0x4-0x6: Reserved 0x7: DISABLED
RESERVED	31:3	0x0	RSV	NA	Reserved

8.2.2.23.17 General Port Configuration - PRTGEN_CNF (0x00083120)

Reset source: LAN Power Good Reset

NVM Load: Type1/2 in LAN Power Good Reset Register Autoload module, Scope: PRT

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PORT_DIS	0	1b	RO	UNDEFINED	Port Disable. Defines if the Ethernet port is enabled from the NVM. Exception: Port 0 is always enabled and cannot be disabled from the NVM. 0b = Enabled. 1b = Disabled.
ALLOW_PORT_DIS	1	0b	RO	UNDEFINED	Allow Port Disable. 0b = Asserting DEV_DIS_N has no effect on this port. 1b = Asserting DEV_DIS_N disables this port.
EMP_PORT_DIS	2	0b	RO	UNDEFINED	Set by the EMP to disable the Ethernet port. The NVM value for this bit should always be 0b (enabled). NVM should use the PORT_DIS bit to disable a port.
RESERVED	31:3	0x0	RSV	NA	

8.2.2.24 Fleep Registers

8.2.2.24.1 Unit Load Status - GLNVM_ULD (0x000B6008)

Reset source: LAN Power Good Reset, NVM Load: Forbidden No Load, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PCIER_DONE	0	0b	RO	NA	The PCIe Reset process is done (all related registers are loaded).
PCIER_DONE_1	1	0b	RO	NA	The PCIe Reset process is done (all related registers are loaded). Mirror of bit 0.
RESERVED	2	0b	RSV	NA	Reserved
MAC0_RESET_DONE	3	0b	RO	NA	The MAC0 Reset process is done (all related registers are loaded).
MAC1_RESET_DONE	4	0b	RO	NA	The MAC1 Reset process is done (all related registers are loaded).
POR_DONE	5	0b	RO	NA	The Power On Reset process is done (all related registers are loaded).
MASTER0_RESET_DONE	6	0b	RO	NA	Reserved
MASTER1_RESET_DONE	7	0b	RO	NA	Reserved
RESERVED	31:8	0x0	RSV	NA	Reserved

8.2.2.24.2 Unit Load Timeout - GLNVM_ULT (0x000B6154)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
CONF_PCIR_AE	0	0b	RW	NA	When set, indicates that the PCIe Autoload Timer has ended before the respective NVM module has been initialized.
CONF_PCIRTL_AE	1	0b	RW	NA	When set, indicates that the PCIe Autoload Timer has ended before the respective NVM module has been initialized.
RESERVED	2	0b	RW	NA	Reserved
CONF_CORE_AE	3	0b	RW	NA	When set, indicates that the General Autoload Timer has ended before the respective NVM module has been initialized.
CONF_GLOBAL_AE	4	0b	RW	NA	When set, indicates that the General Autoload Timer has ended before the respective NVM module has been initialized.
CONF_POR_AE	5	0b	RW	NA	When set, indicates that the General Autoload Timer has ended before the respective NVM module has been initialized.
RESERVED	8:6	000b	RW	NA	Reserved
CONF_PCIALT_AE	9	0b	RW	NA	When set, indicates that the PCIe Autoload Timer has ended before the respective NVM module has been initialized.
CONF_PE_AE	10	0b	RW	NA	When set, indicates that the PCIe Autoload Timer has ended before the respective NVM module has been initialized.
RESERVED	31:11	0x0	RW	NA	Reserved

8.2.2.24.3 Autoload Timers - GLNVM_ALTIMERS (0x000B6140)

Reset source: LAN Power Good Reset, NVM Load: Forbidden Load, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PCI_ALTIMER	11:0	0x010	RW	NA	PCIe Autoload Timer. Default is 16 to 15 ms. resolution is in ms
GEN_ALTIME R	31:12	0x0012C	RW	NA	General Autoload Timer. Default is ~ 300 ms. resolution is in ms.

8.2.2.24.4 Global NVM General Status Register - GLNVM_GENS (0x000B6100)

Reset source: LAN Power Good Reset, NVM Load: Forbidden No Load, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
NVM_PRES	0	0b	RW	NA	NVM Present. Setting this bit to 1b indicates that a flash part is present and that a correct validity field was found in one of the two basic banks (i.e. validity field value read is 01b).
RESERVED	4:1	0x0	RSV	NA	Reserved
SR_SIZE	7:5	0x6	RO	NA	Shadow RAM Size. This field defines the size of the internal shadow RAM. The shadow RAM size is equal to $2^{\text{Field\_Value}}$ KB. Initial value is 110b which correspond to $2^6 = 64$ KB.
BANK1VAL	8	0b	RW	NA	Basic Bank 1 Valid. When set to 1b, indicates that the content of the basic bank 1 of the Flash device is valid. When set to 0b, indicates that the content of basic banks 0 is valid. Meaningful only when NVM_PRES bit is read as 1b. It is written by the hardware once at power-up, and then toggled only by EMP.
RESERVED	31:9	0x0	RSV	NA	

8.2.2.24.5 Flash Access Register - GLNVM_FLA (0x000B6108)

Note: The access type in the PF and VF spaces is determined individually per field. The values are contained in the table that describes the fields in the internal space.

Reset source: LAN Power Good Reset, NVM Load: Forbidden No Load, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	5:0	0x0	RSV	NA	Reserved
LOCKED	6	1b	RW	NA	Normal NVM Programming Mode. When set to 1b, the device is in the normal NVM programming mode. When set to 0b, the device is in the blank flash programming mode. The bit can be cleared by EMP or by SW.
GLOBAL_LOCKED	7	1b	RW	NA	Normal Global NVM Programming Mode. When set to 1b, the device is in the normal Global NVM programming mode. When set to 0b, the device is in the global blank flash programming mode. It is set by FW when detected missing flash descriptor. Once in Blank Flash Programming mode, direct Flash Bit-Bang is possible. The bit can be cleared by EMP or by SW.
RESERVED	31:8	0x0	RSV	NA	Reserved

8.2.2.24.6 Global Status - GLGEN_STAT (0x000B612C)

Reset source: LAN Power Good Reset, NVM Load: Forbidden Load, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	31:0	0x0	RSV	NA	Reserved.

8.2.2.24.7 Protected CSR List - GLNVM_PROTCSR[n] (0x000B6010 + 0x4*n, n=0...59)

Reset source: LAN Power Good Reset, NVM Load: Protected, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
ADDR_BLOCK	23:0	0xFFFFF	RW	NA	CSR Blocked Address. The field contains the address of a 'blocked' register included in the CSR Protected List NVM module. 'Blocked' registers cannot be loaded from a CSR format module (Type 1/2/3). The register is loaded from shadow RAM at POR events only, and only from the CSR Protected List module in NVM. It can be written by EMP. It can be written by host only when in the blank flash programming mode.
RESERVED	31:24	0x0	RSV	NA	Reserved

8.2.2.24.8 General FW Debug Registers - GENERAL_MNG_FW_DBG_CSR[n] (0x000B6180 + 0x4*n, n=0...9)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
GENERAL_FW_DBG	31:0	0x0	RW	NA	All Purpose General FW Debug

8.2.2.24.9 Hardware Arbitration Control - GL_MNG_HWARB_CTRL (0x000B6130)

Reset source: LAN Power Good Reset

NVM Load: Type1/2 in LAN Power Good Reset Register Autoload module, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
NCSI_ARB_EN	0	0b	RW	NA	Hardware Arbitration Enable. If this bit is set, it is assumed the NCSI_ARB_IN and NCSI_ARB_OUT are connected to an Hardware arbitration ring. Otherwise, the NCSI_ARB_IN pin is pulled up internally.
RESERVED	31:1	0x0	RSV	NA	

8.2.2.24.10 Firmware Status - GL_MNG_FWSM (0x000B6134)

This register reflects the Firmware load status. Bits [15:0] in this register are reset by Firmware reset.

Reset source: LAN Power Good Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
FW_MODES	2:0	0x0	RW	NA	Firmware Mode - indicate in which mode the FW operates: 000: Normal Mode 001: Debug mode 010: Recovery Mode 011: Debug + Recovery mode 100: Rollback Mode 101: Rollback + Recovery mode
RESERVED	9:3	0x0	RW	NA	Reserved
EEP_RELOAD_IND	10	0b	RW	NA	NVM reloaded indication Set to 1b after firmware reloads the NVM configuration after a Core reset Cleared by firmware once the first AQ command is received from one of the drivers.
RESERVED	15:11	0x0	RW	NA	Reserved
PCIR_AL_FAILURE	16	0b	RW	NA	PCIR auto-load failure. A PCI Reset is requested
POR_AL_FAILURE	17	0b	RW	NA	POR auto-load failure. A Power on reset is requested
RESERVED	18	0b	RW	NA	Reserved

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
EXT_ERR_IND	24:19	0x0	RW	NA	External error indication Firmware writes here the reason that the firmware operation has stopped. Possible values: 0x0: No error 0x1: Parser module failed 0x2: Switch module failed. 0x3: Scheduler module failed. 0x4: DCB module failed 0x5: Link module failed 0x6: LLDP module failed 0x7: Manage module failed. 0x8: ACL module failed 0x9: Flow director/RSS module failed 0x10-0x3D: Reserved 0x3E: Mini-loader failed (If set, devices is in blank flash programming) 0x3F: ROM process failure (if set, devices is in blank flash programming) Note: Following error detection and GL_MNG_FWSM.EXT_IND_ERR update, the PFINT_ICR0.ADMINQ bit is set and an interrupt is sent to the Host. However when values of 0x0 is placed in this field the PFINT_ICR0.ADMINQ bit is not set and an interrupt is not generated.
RSV5	31:25	0x0	RW	NA	Reserved

8.2.2.25 IOSB SB Registers (imported from CNV)

8.2.2.25.1 SKU Fuses - GL_UFUSE_SOC (0x000A400C)

Reset source: LAN Power Good Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PORT_MODE	1:0	0x1	RO	NA	00 = Octal Port 01 = Quad Port 10 = Dual Port 11 = Single Port
BANDWIDTH_1G	2	0b	RO	NA	When set to 1 limit the speed per port to support maximum 1G bps.
BANDWIDTH_10G	3	0b	RO	NA	When set to 1 limit the speed per port to support maximum 1G bps.
RESERVED	5:4	0x0	RSV	NA	
CSR_PROTECTION_ENABLE	6	0b	RO	NA	0b = CSR Protection disabled 1b = CSR Protection enabled
RESERVED	31:7	0x0	RSV	NA	

8.2.2.26 Diagnostic Registers

The device contains several diagnostic registers. These registers enable software to directly access the contents of the internal Packet Buffer Memory (PBM), also referred to as FIFO space. These registers also give software visibility into other locations in the design that are not for external use.

8.2.2.26.1 global control and status - GLOBAL_FW_CTL (0x0008204C)

Reset source: Management Reset, NVM Load: No, Scope: GL

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
EN_PORT0_CSR_CLK	0	0b	RO	NA	Enable CSR access to LAN port when PHY clock is not valid. (Default is clear)
PHY0_CLK_VALID	1	0b	RO	NA	reflect the valid indication of the PHY port0 clock
EN_PORT1_CSR_CLK	2	0b	RO	NA	Enable CSR access to LAN port when PHY clock is not valid. (Default is clear)
PHY1_CLK_VALID	3	0b	RO	NA	reflect the valid indication of the PHY port0 clock
MNG_CLK_SEL_10N1G_0	4	0b	RO	NA	Clock selection for port 0
AUTO_SWITCH_CLK_STOP_0	5	0b	RO	NA	auto switch on clock stop port 0
MNG_CLK_SEL_10N1G_1	6	0b	RO	NA	Clock selection for port 1
AUTO_SWITCH_CLK_STOP_1	7	0b	RO	NA	auto switch on clock stop port 1
RESERVED	15:8	0x0	RO	NA	RSV
PHY0_CLK_COUNTER_NT	23:16	0x0	RO	NA	8 bits reflect the higher bits of total counter 13 bits of PHY clock 0 (i.e. 5 LSB are not reflected in CSR)
PHY1_CLK_COUNTER_NT	31:24	0x0	RO	NA	8 bits reflect the higher bits of total counter 13 bits of PHY clock 1 (i.e. 5 LSB are not reflected in CSR)

8.2.2.27 VF Registers Mapping in the PF Space

8.2.2.27.1 VF Control Register - VFCTRL[n] (0x00000300 + 0x4*n, n=0...63; WO)

Reset source: Master Reset, NVM Load: No, Scope: PF

Fields definitions are the same as defined on [Section 8.3.3.1.1](#)

8.2.2.27.2 VF Mailbox - VFMAILBOX[n] (0x00004C00 + 0x4*n, n=0...63; RW)

Reset source: Master Reset, NVM Load: No, Scope: PF

Fields definitions are the same as defined on [Section 8.3.3.1.3](#)

8.2.2.27.3 PF Mailbox Memory - PFMBMEM[n,m] (0x00013000 + 0x4*n + 0x40*m, n=0...15, m=0...63; RW)

Reset source: Master Reset, NVM Load: No, Scope: PF

Fields definitions are the same as defined on [Section 8.3.3.1.4](#)

8.2.2.27.4 VF Extended Interrupt Cause - VFEICR[n] (0x00000B00 + 0x4*n, n=0...63; RC/W1C)

Reset source: Master Reset, NVM Load: No, Scope: PF

Fields definitions are the same as defined on [Section 8.3.3.2.1](#)

8.2.2.27.5 VF Extended Interrupt Cause Set - VFEICS[n] (0x00000C00 + 0x4*n, n=0...63; WO)

Reset source: Master Reset, NVM Load: No, Scope: PF

Fields definitions are the same as defined on [Section 8.3.3.2.2](#)

8.2.2.27.6 VF Extended Interrupt Mask Set/Read - VFEIMS[n] (0x00000D00 + 0x4*n, n=0...63; RWS)

Reset source: Master Reset, NVM Load: No, Scope: PF

Fields definitions are the same as defined on [Section 8.3.3.2.3](#)

8.2.2.27.7 VF Extended Interrupt Mask Clear - VFEIMC[n] (0x00000E00 + 0x4*n, n=0...63; WO)

Reset source: Master Reset, NVM Load: No, Scope: PF

Fields definitions are the same as defined on [Section 8.3.3.2.4](#)

8.2.2.27.8 VF Extended Interrupt Auto Mask Enable - VFEIAM[n] (0x00004D00 + 0x4*n, n=0...63; RW)

Reset source: Master Reset, NVM Load: No, Scope: PF

Fields definitions are the same as defined on [Section 8.3.3.2.5](#)

8.2.2.27.9 VF Interrupt Vector Allocation Registers - VFIVAR[n] (0x00012500 + 0x4*n, n=0...63; RW)

Reset source: Master Reset, NVM Load: No, Scope: PF

Fields definitions are the same as defined on [Section 8.3.3.2.7](#)

8.2.2.27.10 VF Interrupt Vector Allocation Registers Misc - VFIVAR_MISC[n] (0x00004E00 + 0x4*n, n=0...63; RW)

Reset source: Master Reset, NVM Load: No, Scope: PF

Fields definitions are the same as defined on [Section 8.3.3.2.8](#)

8.2.2.27.11 VF Multiple Receive Queues Command Register - VFMRQC[n] (0x00003400 + 0x4*n, n=0...63; RW)

Reset source: Master Reset, NVM Load: No, Scope: PF

Fields definitions are the same as defined on [Section 8.3.3.3.9](#)

8.2.2.27.12 VF Good Packets Transmitted Count - VFGPTC[n] (0x00008300 + 0x4*n, n=0...63; RO)

Reset source: Master Reset, NVM Load: No, Scope: PF

Fields definitions are the same as defined on [Section 8.3.3.6.2](#)

8.2.2.27.13 VF Good Octets Transmitted Count LSB - VFGOTC_LSB[n] (0x00008400 + 0x8*n, n=0...63; RO)

Reset source: Master Reset, NVM Load: No, Scope: PF

Fields definitions are the same as defined on [Section 8.3.3.6.5](#)

8.2.2.27.14 VF Good Octets Transmitted Count MSB - VFGOTC_MSB[n] (0x00008404 + 0x8*n, n=0...63; RO)

Reset source: Master Reset, NVM Load: No, Scope: PF

Fields definitions are the same as defined on [Section 8.3.3.6.6](#)

8.2.2.27.15 VF Good Octets Received Count Low - VFGORC_LSB[n] (0x00001020 + 0x40*n, n=0...63; RW)

Reset source: Master Reset, NVM Load: No, Scope: PF

Fields definitions are the same as defined on [Section 8.3.3.6.3](#)

8.2.2.27.16 VF Good Packets Received Count - VFGPRC[n] (0x0000101C + 0x40*n, n=0...63; RW)

Reset source: Master Reset, NVM Load: No, Scope: PF

Fields definitions are the same as defined on [Section 8.3.3.6.1](#)

8.2.2.27.17 VF Good Octets Received Count High - VFGORC_MSB[n] (0x0000D020 + 0x40*n, n=0...63; RW)

Reset source: Master Reset, NVM Load: No, Scope: PF

Fields definitions are the same as defined on [Section 8.3.3.6.4](#)

8.2.2.27.18 VF Multicast Packets Received Count - VFMPRC[n] (0x0000D01C + 0x40*n, n=0...63; RO)

Reset source: Master Reset, NVM Load: No, Scope: PF

Fields definitions are the same as defined on [Section 8.3.3.6.7](#)

8.3 Device Registers - VF

8.3.1 VF Registers Mapping in the PF Space

Table 8-5. VF Registers Mapping in the PF Space

Abbreviation	Virtual Address	Physical Address
VFSRRCTL	0x00001014 + 0x40*n, n=0...7	0x00002100 + 0x4*n, n=0...15
VFTPH_RXCTRL	0x0000100C + 0x40*n, n=0...7	0x00002200 + 0x4*n, n=0...15

8.3.2 BAR0 Registers Summary

Table 8-6. BAR0 Registers Summary

Offset / Alias Offset	Abbreviation	Name	Section
General Control Registers - VF			
0x00000000	VFCTRL	VF Control Register	Section 8.3.3.1.1
0x00000008	VFSTATUS	VF Device Status Register	Section 8.3.3.1.2
0x000002FC	VFMAILBOX	VF Mailbox	Section 8.3.3.1.3
0x00000200 + 0x4*n, n=0...15	VFMBMEM[n]	VF Mailbox Memory	Section 8.3.3.1.4
0x00000010	VFLINKS	Link Status Register	Section 8.3.3.1.5
Interrupt Registers - VF			
0x00000100	VFEICR	VF Extended Interrupt Cause	Section 8.3.3.2.1
0x00000104	VFEICS	VF Extended Interrupt Cause Set	Section 8.3.3.2.2
0x00000108	VFEIMS	VF Extended Interrupt Mask Set/ Read	Section 8.3.3.2.3
0x0000010C	VFEIMC	VF Extended Interrupt Mask Clear	Section 8.3.3.2.4
0x00000114	VFEIAM	VF Extended Interrupt Auto Mask Enable	Section 8.3.3.2.5
0x00000820 + 0x4*n, n=0...1	VFEITR[n]	VF Extended Interrupt Throttle Registers	Section 8.3.3.2.6
0x00000120 + 0x4*n, n=0...3	VFIVAR[n]	VF Interrupt Vector Allocation Registers	Section 8.3.3.2.7
0x00000140	VFIVAR_MISC	VF Interrupt Vector Allocation Registers Misc	Section 8.3.3.2.8
0x00000148	VFPBACL	VF MSI-X PBA Clear	Section 8.3.3.2.9

Table 8-6. BAR0 Registers Summary [continued]

Offset / Alias Offset	Abbreviation	Name	Section
Receive Registers - VF			
0x00001000 + 0x40*n, n=0...7	VFRDBAL[n]	VF Receive Descriptor Base Address Low	Section 8.3.3.3.1
0x00001004 + 0x40*n, n=0...7	VFRDBAH[n]	VF Receive Descriptor Base Address High	Section 8.3.3.3.2
0x00001008 + 0x40*n, n=0...7	VFRDLEN[n]	VF Receive Descriptor Length	Section 8.3.3.3.3
0x00001010 + 0x40*n, n=0...7	VFRDH[n]	VF Receive Descriptor Head	Section 8.3.3.3.4
0x00001018 + 0x40*n, n=0...7	VFRDT[n]	VF Receive Descriptor Tail	Section 8.3.3.3.5
0x00001028 + 0x40*n, n=0...7	VFRXDCTL[n]	VF Receive Descriptor Control	Section 8.3.3.3.6
0x00001014 + 0x40*n, n=0...7	VFSRRCTL[n]	VF Split and Replication Receive Control Registers	Section 8.3.3.3.7
0x00000300	VFPSRTYPE	VF Replication Packet Split Receive Type Register	Section 8.3.3.3.8
0x00003000	VFMRQC	VF Multiple Receive Queues Command Register	Section 8.3.3.3.9
0x00003100 + 0x4*n, n=0...9	VFRSSRK[n]	VF RSS Random Key Register	Section 8.3.3.3.10
0x00003200 + 0x4*n, n=0...15	VFRETA[n]	VF Redirection Table	Section 8.3.3.3.11
Transmit Registers - VF			
0x00002000 + 0x40*n, n=0...7	VFTDBAL[n]	VF Transmit Descriptor Base Address Low	Section 8.3.3.4.1
0x00002004 + 0x40*n, n=0...7	VFTDBAH[n]	VF Transmit Descriptor Base Address High	Section 8.3.3.4.2
0x00002008 + 0x40*n, n=0...7	VFTDLEN[n]	VF Transmit Descriptor Length	Section 8.3.3.4.3
0x00002010 + 0x40*n, n=0...7	VFTDH[n]	VF Transmit Descriptor Head	Section 8.3.3.4.4
0x00002018 + 0x40*n, n=0...7	VFTDT[n]	VF Transmit Descriptor Tail	Section 8.3.3.4.5
0x00002028 + 0x40*n, n=0...7	VFTXDCTL[n]	VF Transmit Descriptor Control	Section 8.3.3.4.6
0x00002038 + 0x40*n, n=0...7	VFTDWBAL[n]	VF Tx Descriptor Completion Write Back Address Low	Section 8.3.3.4.7
0x0000203C + 0x40*n, n=0...7	VFTDWBAH[n]	VF Tx Descriptor Completion Write Back Address High	Section 8.3.3.4.8
TPH Registers - VF			
0x0000100C + 0x40*n, n=0...7	VFTPH_RXCTRL[n]	VF Rx TPH Control Register	Section 8.3.3.5.1
0x0000200C + 0x40*n, n=0...7	VFTPH_TXCTRL[n]	VF Tx TPH Control Registers	Section 8.3.3.5.2

Table 8-6. BAR0 Registers Summary [continued]

Offset / Alias Offset	Abbreviation	Name	Section
Statistic Register Descriptions - VF			
0x0000101C	VFGPRC	VF Good Packets Received Count	Section 8.3.3.6.1
0x0000201C	VFGPTC	VF Good Packets Transmitted Count	Section 8.3.3.6.2
0x00001020	VFGORC_LSB	VF Good Octets Received Count Low	Section 8.3.3.6.3
0x00001024	VFGORC_MSB	VF Good Octets Received Count High	Section 8.3.3.6.4
0x00002020	VFGOTC_LSB	VF Good Octets Transmitted Count LSB	Section 8.3.3.6.5
0x00002024	VFGOTC_MSB	VF Good Octets Transmitted Count MSB	Section 8.3.3.6.6
0x00001034	VFMPRC	VF Multicast Packets Received Count	Section 8.3.3.6.7

8.3.3 Detailed Register Description - VF BAR0

8.3.3.1 General Control Registers - VF

8.3.3.1.1 VF Control Register - VFCTRL (0x00000000)

Reset source: Master Reset, NVM Load: No, Scope: VF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RESERVED	25:0	0x0	RSV	NA	Reserved.
RST	26	0b	WO	NA	VF Reset. This bit performs a reset of the queue enable and the interrupt registers of the VF.
RESERVED	31:27	0x0	RSV	NA	Reserved.

8.3.3.1.2 VF Device Status Register - VFSTATUS (0x00000008; RO)

Reset source: Master Reset, NVM Load: No, Scope: VF

Fields definitions are the same as defined on [Section 8.2.2.1.2](#)

8.3.3.1.3 VF Mailbox - VFMAILBOX (0x000002FC)

Reset source: Master Reset, NVM Load: No, Scope: VF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
REQ	0	0b	WO	NA	Request for PF Ready. Setting this bit, causes an interrupt to the PF. This bit always reads as 0b. Setting this bit sets the corresponding bit in the VFREQ field in PFMBICR register.
ACK	1	0b	WO	NA	PF message received. Setting this bit, causes an interrupt to the PF. This bit always reads as 0b. Setting this bit sets the corresponding bit in the VFACK field in PFMBICR register.
VFU	2	0b	RW	NA	Buffer is Taken by VF. This bit can be set only if the PFU bit is cleared and is mirrored in the VFU bit of the PFMailbox register.
PFU	3	0b	RW	NA	Buffer is taken by PF. This bit is RO for the VF and is a mirror of the PFU bit of the PFMailbox register.
PFSTS	4	0b	RC	NA	PF wrote a message in the mailbox.
PFACK	5	0b	RC	NA	PF acknowledged the VF previous message.
RSTI	6	1b	RO	NA	Indicates that the PF reset the shared resources and the reset sequence is in progress. This bit is not affected by VFLR.
RSTD	7	0b	RC	NA	Indicates that a PF software reset completed.
RESERVED	31:8	0x0	RSV	NA	Reserved.

8.3.3.1.4 VF Mailbox Memory - VFMBMEM[n] (0x00000200 + 0x4*n, n=0...15)

Reset source: Master Reset, NVM Load: No, Scope: VF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MAILBOX_DATA	31:0	X	RW	NA	Mailbox data is composed of 16 x 4 byte registers.

8.3.3.1.5 Link Status Register - VFLINKS (0x00000010; RO)

Reset source: MAC Reset, NVM Load: No, Scope: VF

Fields definitions are the same as defined on [Section 8.2.2.15.8](#)

8.3.3.2 Interrupt Registers - VF

8.3.3.2.1 VF Extended Interrupt Cause - VFEICR (0x00000100)

Reset source: Master Reset, NVM Load: No, Scope: VF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MSIX	2:0	0x0	RW1C	DYNAMIC	Indicates an interrupt cause mapped to MSI-X vectors 2:0.
RESERVED	31:3	0x0	RSV	NA	Reserved.

8.3.3.2.2 VF Extended Interrupt Cause Set - VFEICS (0x00000104)

Reset source: Master Reset, NVM Load: No, Scope: VF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MSIX	2:0	0x0	WO	NA	Sets to the corresponding EICR bit of MSI-X vectors 2:0.
RESERVED	31:3	0x0	RSV	NA	Reserved.

8.3.3.2.3 VF Extended Interrupt Mask Set/Read - VFEIMS (0x00000108)

Reset source: Master Reset, NVM Load: No, Scope: VF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MSIX	2:0	0x0	RWS	NA	Sets the Mask bit for the corresponding EICR bit of MSI-X vectors 2:0.
RESERVED	31:3	0x0	RSV	NA	Reserved.

8.3.3.2.4 VF Extended Interrupt Mask Clear - VFEIMC (0x0000010C)

Reset source: Master Reset, NVM Load: No, Scope: VF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MSIX	2:0	0x0	WO	NA	Clears the Mask bit for the corresponding EICR bit of MSI-X vectors 2:0.
RESERVED	31:3	0x0	RSV	NA	Reserved.

8.3.3.2.5 VF Extended Interrupt Auto Mask Enable - VFEIAM (0x00000114)

Reset source: Master Reset, NVM Load: No, Scope: VF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MSIX	2:0	0x0	RW	NA	Auto Mask bit for the corresponding EICR bit of MSI-X vectors 2:0.
RESERVED	31:3	0x0	RSV	NA	Reserved.

8.3.3.2.6 VF Extended Interrupt Throttle Registers - VFEITR[n] (0x00000820 + 0x4*n, n=0...1; RW)

Reset source: Master Reset, NVM Load: No, Scope: VF

Fields definitions are the same as defined on [Section 8.2.2.6.16](#)

8.3.3.2.7 VF Interrupt Vector Allocation Registers - VFIVAR[n] (0x00000120 + 0x4*n, n=0...3)

Reset source: Master Reset, NVM Load: No, Scope: VF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
INT_ALLOC_0	0	0b	RW	NA	Defines the MSI-X vector (0 or 1) assigned to Rx queue '2*N' for IVAR 'N' register (N=0...3).
RESERVED	6:1	0x0	RSV	NA	Reserved.
INT_ALLOC_V AL_0	7	0b	RW	NA	Valid bit for INT_Alloc[0].
INT_ALLOC_1	8	0b	RW	NA	Defines the MSI-X vector (0 or 1) assigned to Tx queue '2*N' for IVAR 'N' register (N=0...3).
RESERVED	14:9	0x0	RSV	NA	Reserved.
INT_ALLOC_V AL_1	15	0b	RW	NA	Valid bit for INT_Alloc[1].
INT_ALLOC_2	16	0b	RW	NA	Defines the MSI-X vector (0 or 1) assigned to Rx queue '2*N+1' for IVAR 'N' register (N=0...3).
RESERVED	22:17	0x0	RSV	NA	Reserved.
INT_ALLOC_V AL_2	23	0b	RW	NA	Valid bit for INT_Alloc[2].
INT_ALLOC_3	24	0b	RW	NA	Defines the MSI-X vector (0 or 1) assigned to Tx queue '2*N+1' for IVAR 'N' register (N=0...3).
RESERVED	30:25	0x0	RSV	NA	Reserved.
INT_ALLOC_V AL_3	31	0b	RW	NA	Valid bit for INT_Alloc[3].

8.3.3.2.8 VF Interrupt Vector Allocation Registers Misc - VFIVAR_MISC (0x00000140)

Reset source: Master Reset, NVM Load: No, Scope: VF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
INT_ALLOC_0	1:0	X	RW	NA	Defines the MSI-X vector assigned to the mailbox interrupt.
RESERVED	6:2	0x0	RSV	NA	Reserved.
INT_ALLOC_V AL_0	7	0b	RW	NA	Valid bit for INT_Alloc[0].
RESERVED	31:8	0x0	RSV	NA	Reserved.

8.3.3.2.9 VF MSI-X PBA Clear - VFPBACL (0x00000148)

Reset source: PCIe In-band Reset, NVM Load: No, Scope: VF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PENBIT	2:0	0x0	RW1C	DYNAMIC	MSI-X Pending Bits Clear. Writing a 1b to any bit clears the corresponding MSIXPBA bit; writing a 0b has no effect. Reading this register returns the PBA vector.
RESERVED	31:3	0x0	RSV	NA	Reserved.

8.3.3.3 Receive Registers - VF

8.3.3.3.1 VF Receive Descriptor Base Address Low - VFRDBAL[n] (0x00001000 + 0x40*n, n=0...7; RW)

Reset source: Master Reset, NVM Load: No, Scope: VF

Fields definitions are the same as defined on [Section 8.2.2.9.1](#)

8.3.3.3.2 VF Receive Descriptor Base Address High - VFRDBAH[n] (0x00001004 + 0x40*n, n=0...7; RW)

Reset source: Master Reset, NVM Load: No, Scope: VF

Fields definitions are the same as defined on [Section 8.2.2.9.2](#)

8.3.3.3.3 VF Receive Descriptor Length - VFRDLEN[n] (0x00001008 + 0x40*n, n=0...7; RW)

Reset source: Master Reset, NVM Load: No, Scope: VF

Fields definitions are the same as defined on [Section 8.2.2.9.3](#)

8.3.3.3.4 VF Receive Descriptor Head - VFRDH[n] (0x00001010 + 0x40*n, n=0...7; RO)

Reset source: Master Reset, NVM Load: No, Scope: VF

Fields definitions are the same as defined on [Section 8.2.2.9.4](#)

8.3.3.3.5 VF Receive Descriptor Tail - VFRDT[n] (0x00001018 + 0x40*n, n=0...7; RW)

Reset source: Master Reset, NVM Load: No, Scope: VF

Fields definitions are the same as defined on [Section 8.2.2.9.5](#)

8.3.3.3.6 VF Receive Descriptor Control - VFRXDCTL[n] (0x00001028 + 0x40*n, n=0...7; RW)

Reset source: Master Reset, NVM Load: No, Scope: VF

Fields definitions are the same as defined on [Section 8.2.2.9.6](#)

8.3.3.3.7 VF Split and Replication Receive Control Registers - VFSRRCTL[n] (0x00001014 + 0x40*n, n=0...7; RW)

Reset source: Master Reset, NVM Load: No, Scope: VF

Fields definitions are the same as defined on [Section 8.2.2.9.7](#)

8.3.3.3.8 VF Replication Packet Split Receive Type Register - VFPSRTYPE (0x00000300; RW)

Reset source: Master Reset, NVM Load: No, Scope: VF

Fields definitions are the same as defined on [Section 8.2.2.8.4](#)

8.3.3.3.9 VF Multiple Receive Queues Command Register - VFMRQC (0x00003000)

Reset source: Master Reset, NVM Load: No, Scope: VF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
RSSE	0	0b	RW	NA	RSS Enable. This bit enables the VF RSS mechanism. 0b = VF RSS disabled. 1b = VF RSS enable.
RESERVED	15:1	0x0	RSV	NA	Reserved.
RSS_FIELD_ENABLE	31:16	0x0	RW	NA	Each bit, when set, enables a specific field selection to be used by the hash function. Several bits can be set at the same time. Bit[16] - Enable TcpIPv4 hash function. Bit[17] - Enable IPv4 hash function. Bit[18] - Reserved. Bit[19] - Reserved. Bit[20] - Enable IPv6 hash function. Bit[21] - Enable TcpIPv6 hash function. Bit[22] - Enable UdpIPv4. Bit[23] - Enable UdpIPv6. Bit[24] - Reserved. Bits[31:25] - Reserved; 0x0. Note: On Tunnel packets IPv4-IPv6, only the IPv4 header might be used for the RSS filtering.

8.3.3.3.10 VF RSS Random Key Register - VFRSSRK[n] (0x00003100 + 0x4*n, n=0...9; RW)

Reset source: Master Reset, NVM Load: No, Scope: VF

Fields definitions are the same as defined on [Section 8.2.2.8.20](#)

8.3.3.3.11 VF Redirection Table - VFRETA[n] (0x00003200 + 0x4*n, n=0...15; RW)

Reset source: Master Reset, NVM Load: No, Scope: VF

Fields definitions are the same as defined on [Section 8.2.2.8.21](#)

8.3.3.4 Transmit Registers - VF

8.3.3.4.1 VF Transmit Descriptor Base Address Low - VFTDBAL[n] (0x00002000 + 0x40*n, n=0...7; RW)

Reset source: Master Reset, NVM Load: No, Scope: VF

Fields definitions are the same as defined on [Section 8.2.2.10.5](#)

8.3.3.4.2 VF Transmit Descriptor Base Address High - VFTDBAH[n] (0x00002004 + 0x40*n, n=0...7; RW)

Reset source: Master Reset, NVM Load: No, Scope: VF

Fields definitions are the same as defined on [Section 8.2.2.10.6](#)

8.3.3.4.3 VF Transmit Descriptor Length - VFTDLEN[n] (0x00002008 + 0x40*n, n=0...7; RW)

Reset source: Master Reset, NVM Load: No, Scope: VF

Fields definitions are the same as defined on [Section 8.2.2.10.7](#)

8.3.3.4.4 VF Transmit Descriptor Head - VFTDH[n] (0x00002010 + 0x40*n, n=0...7; RO)

Reset source: Master Reset, NVM Load: No, Scope: VF

Fields definitions are the same as defined on [Section 8.2.2.10.8](#)

8.3.3.4.5 VF Transmit Descriptor Tail - VFTDT[n] (0x00002018 + 0x40*n, n=0...7; RW)

Reset source: Master Reset, NVM Load: No, Scope: VF

Fields definitions are the same as defined on [Section 8.2.2.10.9](#)

8.3.3.4.6 VF Transmit Descriptor Control - VFTXDCTL[n] (0x00002028 + 0x40*n, n=0...7; RW)

Reset source: Master Reset, NVM Load: No, Scope: VF

Fields definitions are the same as defined on [Section 8.2.2.10.10](#)

8.3.3.4.7 VF Tx Descriptor Completion Write Back Address Low - VFTDWBAL[n] (0x00002038 + 0x40*n, n=0...7; RW)

Reset source: Master Reset, NVM Load: No, Scope: VF

Fields definitions are the same as defined on [Section 8.2.2.10.11](#)

8.3.3.4.8 VF Tx Descriptor Completion Write Back Address High - VFTDWBAH[n] (0x0000203C + 0x40*n, n=0...7; RW)

Reset source: Master Reset, NVM Load: No, Scope: VF

Fields definitions are the same as defined on [Section 8.2.2.10.12](#)

8.3.3.5 TPH Registers - VF

8.3.3.5.1 VF Rx TPH Control Register - VFTPH_RXCTRL[n] (0x0000100C + 0x40*n, n=0...7; RW)

Reset source: Master Reset, NVM Load: No, Scope: VF

Fields definitions are the same as defined on [Section 8.2.2.12.2](#)

8.3.3.5.2 VF Tx TPH Control Registers - VFTPH_TXCTRL[n] (0x0000200C + 0x40*n, n=0...7; RW)

Reset source: Master Reset, NVM Load: No, Scope: VF

Fields definitions are the same as defined on [Section 8.2.2.12.3](#)

8.3.3.6 Statistic Register Descriptions - VF

Registers in this section are RO by VF and RW by PF. Statistics are reset by PF clearing the register.

8.3.3.6.1 VF Good Packets Received Count - VFGPRC (0x0000101C)

Reset source: Master Reset, NVM Load: No, Scope: VF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VFGPRC	31:0	0x0	RW	NA	Number of good packets received for this VF (of any length). This counter includes loopback packets or replications of multicast packets. The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000.

8.3.3.6.2 VF Good Packets Transmitted Count - VFGPTC (0x0000201C)

Reset source: Master Reset, NVM Load: No, Scope: VF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VFGPTC	31:0	0x0	RO	NA	Number of good packets sent by the queues allocated to this VF. A packet is considered as transmitted if it was forwarded to the MAC unit for transmission to the network and/or is accepted by the internal Tx to Rx switch enablement logic. Packets dropped due to anti-spoofing filtering or loopback packets that are rejected by the Tx to Rx switch are not counted. The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000.

8.3.3.6.3 VF Good Octets Received Count Low - VFGORC_LSB (0x00001020)

Reset source: Master Reset, NVM Load: No, Scope: VF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VFGORC_LSB	31:0	0x0	RW	NA	Number of good octets received (32 LSB of a 36-bit counter) by the queues allocated to this VF. The counter includes loopback packets or replications of multicast packets. This register includes bytes received in a packet from the <Destination Address> field through the <CRC> field, inclusively. Octets are counted on the VF interface rather than on the network interface. The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000.

8.3.3.6.4 VF Good Octets Received Count High - VFGORC_MSB (0x00001024)

Reset source: Master Reset, NVM Load: No, Scope: VF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VFGORC_MSB	3:0	0x0	RW	NA	Number of good octets received (4 MSB of a 36-bit counter) by the queues allocated to this VF. See the VFGORC_LSB register description for more details. The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xF to 0x0.
RESERVED	31:4	0x0	RSV	NA	Reserved.

8.3.3.6.5 VF Good Octets Transmitted Count LSB - VFGOTC_LSB (0x00002020)

Reset source: Master Reset, NVM Load: No, Scope: VF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VFGOTC_LSB	31:0	0x0	RO	NA	Number of good octets transmitted (32 LSB of a 36-bit counter) by the queues allocated to this VF. This register includes bytes transmitted in a packet from the <Destination Address> field through the <CRC> field, inclusively. This register counts octets of the packets counted by the VFGPTC register. Octets are counted on the VF interface rather than on the network interface. The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000.

8.3.3.6.6 VF Good Octets Transmitted Count MSB - VFGOTC_MSB (0x00002024)

Reset source: Master Reset, NVM Load: No, Scope: VF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VFGOTC_MSB	3:0	0x0	RO	NA	Number of good octets transmitted (4 MSB of a 36-bit counter) by the queues allocated to this VF. See the description of the VFGOTC-LSB register for more details. The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xF to 0x0.
RESERVED	31:4	0x0	RSV	NA	Reserved.

8.3.3.6.7 VF Multicast Packets Received Count - VFMPRC (0x00001034)

Reset source: Master Reset, NVM Load: No, Scope: VF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
VFMPRC	31:0	0x0	RO	NA	Number of multicast good packets received by this VF (of any length) that pass the Ethernet MAC Address filtering (excluding broadcast packets). The counter does not count received flow control packets. This register increments only if receives are enabled. This register does not count packets counted by the Missed Packet Count (MPC) register. This counter also includes loopback packets or replications of multicast packets. The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000.

8.3.4 BAR4 Registers Summary

Table 8-7. BAR4 Registers Summary

Offset / Alias Offset	Abbreviation	Name	Section
MSI-X Register VF			
0x00000000 + 0x10*n, n=0...2	VFMSIXTADD[n]	MSI-X Table Entry Lower Address	Section 8.3.5.1.1
0x00000004 + 0x10*n, n=0...2	VFMSIXTUADD[n]	MSI-X Table Entry Upper Address	Section 8.3.5.1.2
0x00000008 + 0x10*n, n=0...2	VFMSIXMSG[n]	MSI-X Table Entry Message	Section 8.3.5.1.3
0x0000000C + 0x10*n, n=0...2	VFMSIXCTRL[n]	MSI-X Table Entry Vector Control	Section 8.3.5.1.4
0x00002000	VFMSIXPBA	MSIXPBA Bit Description	Section 8.3.5.1.5

8.3.5 Detailed Register Description - VF BAR4

8.3.5.1 MSI-X Register VF

8.3.5.1.1 MSI-X Table Entry Lower Address - VFMSIXTADD[n] (0x00000000 + 0x10*n, n=0...2)

MSI_X Address - lower part.

Reset source: PCIe In-band Reset, NVM Load: No, Scope: VF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MSIXTADD_10	1:0	0x0	RW	NA	Message Address. For proper DWord alignment, software must always write zeros to these two bits; otherwise, the result is undefined. The state of these bits after reset must be 0b. These bits are permitted to be read-only or read/write.
MSIXTADD	31:2	0x0	RW	NA	Message Address. System-specified message lower address. For MSI-X messages, the contents of this field from an MSI-X table entry specifies the lower portion of the DWord-aligned address (AD[31:02]) for the memory write transaction. This field is read/write.

8.3.5.1.2 MSI-X Table Entry Upper Address - VFMSIXTUADD[n] (0x00000004 + 0x10*n, n=0...2)

MSI-X Address - upper part.

Reset source: PCIe In-band Reset, NVM Load: No, Scope: VF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MSIXTUADD	31:0	0x0	RW	NA	Message Upper Address. System-specified message upper address bits. If this field is zero, Single Address Cycle (SAC) messages are used. If this field is non-zero, Dual Address Cycle (DAC) messages are used. This field is read/write.

8.3.5.1.3 MSI-X Table Entry Message - VFMSIXTMSG[n] (0x00000008 + 0x10*n, n=0...2)

MSI-X Data.

Reset source: PCIe In-band Reset, NVM Load: No, Scope: VF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MSIXTMSG	31:0	0x0	RW	NA	Message Data. System-specified message data. For MSI-X messages, the contents of this field from an MSI-X table entry specifies the data driven on AD[31:0] during the memory write transaction's data phase. This field is read/write.

8.3.5.1.4 MSI-X Table Entry Vector Control - VFMSIXVCTRL[n] (0x0000000C + 0x10*n, n=0...2)

MSI-X Vector control.

Reset source: PCIe In-band Reset, NVM Load: No, Scope: VF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
MSIXVCTRL	0	1b	RW	NA	Mask Bit. When this bit is set, the function is prohibited from sending a message using this MSI-X table entry. However, any other MSI-X table entries programmed with the same vector are still capable of sending an equivalent message unless they are also masked. This bit's state after reset is 1b (entry is masked). This bit is read/write.
RESERVED	31:1	0x0	RSV	NA	Reserved. After reset, the state of these bits must be 0b. However, for potential future use, software must preserve the value of these reserved bits when modifying the value of other Vector Control bits. If software modifies the value of these reserved bits, the result is undefined.

8.3.5.1.5 MSIXPBA Bit Description - VFMSIXPBA (0x00002000)

MSI-X Pending bit array.

Reset source: PCIe In-band Reset, NVM Load: No, Scope: VF

Field	Bit(s)	Init.	Access Type	CFG Policy	Description
PENDING_BITS	2:0	0x0	RO	NA	For each pending bit that is set, the function has a pending message for the associated MSI-X Table entry. Pending bits that have no associated MSI-X table entry are reserved.
RESERVED	31:3	0x0	RSV	NA	Reserved.

Note: If a page size larger than 8K is programmed in the IOV structure, the address of the MSIX PBA table will move to be page aligned.

NOTE: *This page intentionally left blank.*

Chapter 9 PCIe Programming Interface

9.1 Overview

The E610 is a multi-function device with the following functions:

- LAN 0
- LAN 1

Different parameters affect how LAN functions are exposed on PCIe.

Both functions contain the following regions of the PCI configuration space (some of them are enabled by NVM settings as detailed in the following sections):

- Mandatory PCI configuration registers - [Section 9.2.2](#)
- Legacy PCI capabilities - [Section 9.2.3](#)
 - Power management capabilities
 - MSI / MSI-X capabilities
 - Vital Product Data (VPD) capability
- PCIe extended capabilities - [Section 9.2.4](#)
 - Advanced Error Reporting (AER)
 - Serial ID
 - Alternate requester ID (ARI).
 - Single root I/O Virtualization (SR-IOV)
 - Latency Tolerance Reporting
 - TPH Requester
 - Access Control Services (ACS)
 - Secondary PCI Express
 - Data Object Exchange Capability (DOE)

9.1.1 Register Attributes

Table 9-1 lists the register attributes used in this section.

Table 9-1. Register Attribute Descriptions

Type	Description
RO	Read-Only register Register bits are read-only and cannot be altered by software.
RW	Read/Write register Register bits are read-write and can be either set or reset.
RW1C	Read-only status — Write 1b to Clear status register Writing a 0b to RW1C bits has no effect.
ROS	Read-Only register with Sticky bits Register bits are read-only and cannot be altered by software. Bits are not cleared by reset and can only be reset with the PWRGOOD signal. Devices that consume AUX power are not allowed to reset sticky bits when AUX power consumption (either via AUX power or PME Enable) is enabled.
RWS	Read/Write register with Sticky bits Register bits are read-write and can be either set or reset by software to the desired state. Bits are not cleared by reset and can only be reset with the PWRGOOD signal. Devices that consume AUX power are not allowed to reset sticky bits when AUX power consumption (either via AUX power or PME Enable) is enabled.
RW1CS	Read-only status — Write 1b to Clear status register Register bits indicate status when read, a set bit, indicating a status event, can be cleared by writing a 1b to it. Writing a 0b to RW1C bits has no effect. Bits are not cleared by reset and can only be reset with the PWRGOOD signal. Devices that consume AUX power are not allowed to reset sticky bits when AUX power consumption (either via AUX power or PME Enable) is enabled.
HwInit	Hardware Initialized Register bits are initialized by firmware or hardware mechanisms such as pin strapping or serial NVM. Bits are read-only after initialization and can only be reset (for write-once by firmware) with the PWRGOOD signal.
RsvdP	Reserved and Preserved Reserved for future read/write implementations; software must preserve value read for writes to these bits.
RsvdZ	Reserved and Zero Reserved for future RW1C implementations; software must use 0b for writes to these bits.

9.2 PCIe Register Map

9.2.1 PCIe Configuration Space Summary

Table 9-2 lists the PCIe configuration registers while their detailed description is given in the sections that follow. PCI configuration fields in the summary table are presented by the following marking:

- Fields that have meaningful default values are indicated in parenthesis — (**value**).
- Dotted fields indicates the same value for both LAN functions
- Light-blue fields indicate read-only fields (loaded from the NVM)
- Magenta fields indicate hard-coded values.
- Other fields contain RW attributes.

Table 9-2. PCI Configuration Registers Map - LAN Functions

Section	Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
Mandatory PCI Register	0x0	Device ID		Vendor ID	
	0x4	Status Register		Command Register	
	0x8	Class Code (0x020000/0x010000)			Revision ID
	0xC	Reserved	Header Type (0x0/0x80)	Latency Timer	Cache Line Size (0x0)
	0x10	Base Address Register 0			
	0x14	Base Address Register 1			
	0x18	Base Address Register 2			
	0x1C	Base Address Register 3			
	0x20	Base Address Register 4			
	0x24	Base Address Register 5			
	0x28	CardBus CIS pointer (0x0000)			
	0x2C	Subsystem ID		Subsystem Vendor ID	
	0x30	Expansion ROM Base Address			
	0x34	Reserved			Cap Ptr (0x40)
	0x38	Reserved			
	0x3C	Max Latency (0x00)	Min Grant (0x00)	Interrupt Pin (0x01...0x04)	Interrupt Line (0x00)
PCI / PCIe Capabilities	0x40...0x4F	Power Management Capability			
	0x50...0x6F	MSI Capability			
	0xB0...0xCF	MSI-X Capability			
	0x70...0xAF	PCIe Capability			
	0xD0...0xDF	VPD Capability			

Table 9-2. PCI Configuration Registers Map - LAN Functions [continued]

Section	Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
Extended PCIe Configuration	0x100...0x144	AER Capability			
	0x148...0x150	Serial ID Capability			
	0x158...0x15C	ARI Capability			
	0x1C4...0x200	SR-IOV Capability			
	0x204...0x28C	TPH Capability			
	0x290...0x294	ACS Capability			
	0x298...0x29C	LTR Capability			
	0x168...0x194	Secondary PCI Express Extended Capability			
	0x3D8...0x3E0	Data Link Feature			
	0x188...0x1AC	Physical Layer 16.0 GT/s Extended Capability			
	0x1AC...0x1C0	Lane Margining at the Receiver Capability			
	0x2A0...0x2A4	RAS DES Capability (VSEC)			
	0x3A0...0x3A4	RAS Data Path Protection Capability (VSEC)			
	0x580...0x594	Data Object Exchange (DOE) Capability			

Table 9-3. PCIe Configuration Registers Map - Dummy Function

Section	Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
Mandatory PCI Register	0x0	Device ID		Vendor ID	
	0x4	Status Register		Command Register	
	0x8	Class Code (0xFF0000)			Revision ID
	0xC	Reserved	Header Type (0x0/0x80)	Latency Timer	Cache Line Size (0x0)
	0x10	Base Address Register 0			
	0x14	Base Address Register 1 (0x0 in 32-bit BAR Mode)			
	0x18	Base Address Register 2 (0x0)			
	0x1C	Base Address Register 3 (0x0)			
	0x20	Base Address Register 4 (0x0)			
	0x24	Base Address Register 5 (0x0)			
	0x28	CardBus CIS pointer (0x0000)			
	0x2C	Subsystem Device ID		Subsystem Vendor ID	
	0x30	Expansion ROM Base Address (0x0)			
	0x34	Reserved			Cap Ptr (0x40)
	0x38	Reserved			
	0x3C	Max Latency (0x00)	Min Grant (0x00)	Interrupt Pin (0x00)	Interrupt Line (0x00)
PCI / PCIe Capabilities	0x40...0x47	Power Management Capability			
	0x70...0xAF	PCIe Capability			

Table 9-3. PCIe Configuration Registers Map [continued]- Dummy Function [continued]

Section	Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
Extended PCIe Configuration	0x100...0x144	AER Capability			
	0x148...0x150	Serial ID Capability			
	0x158...0x15C	ARI Capability			
	0x290...0x294	ACS Capability			
	0x298...0x29C	LTR Capability			
	0x168...0x194	Secondary PCI Express Extended Capability			
	0x3D8...0x3E0	Data Link Feature			
	0x188...0x1AC	Physical Layer 16.0 GT/s Extended Capability			
	0x1AC...0x1C0	Lane Margining at the Receiver Capability			
	0x2A0...0x2A4	RAS DES Capability (VSEC)			
	0x3A0...0x3A4	RAS Data Path Protection Capability (VSEC)			
	0x580...0x594	Data Object Exchange (DOE) Capability			

9.2.1.1 Sharing Among PCI Functions

The E610 supports multiple PCI functions. As each function exposes a PCIe configuration space, each register and each field is either shared among the functions or is replicated per each PCI function. In each section a table summarizes configuration sharing of the registers described in this section. Also, the description of each field describes special considerations regarding configuration sharing.

9.2.2 Mandatory PCI Configuration Registers

Table 9-4. Configuration Sharing of PCI Configuration Space

Field	Sub-field	Shared?	Replicated?	Comments
Vendor ID	Vendor ID	x		
Device ID	Device ID		x	
Command Register	I/O Access Enable		x	Issue UR per PF if disabled.
	Memory Access Enable		x	Issue UR per PF if disabled.
	Bus Master Enable		x	
	Parity Error Response		x	Enables certain error reporting per PF.
	SERR# Enable		x	Controls error reporting per PF.
	Interrupt Disable		x	Selection of interrupt method per PF.

Table 9-4. Configuration Sharing of PCI Configuration Space [continued]

Field	Sub-field	Shared?	Replicated?	Comments
Status Register	Interrupt Status		x	
	Capabilities List	x		Hard-wired to 1b.
	Data Parity Reported / Master Data Parity Error		x	Reports poisoned packets per PF.
	Signaled Target Abort		x	Reports Completer Abort per PF.
	Received Target Abort		x	Reports receiving a Completer Abort per PF.
	Received Master Abort		x	Reports receiving an UR per PF.
	Signaled System Error		x	Reports Fatal / non-fatal message per PF.
	Detected Parity Error		x	Reports receiving a poisoned TLP per PF.
Revision Register		x		
Class Code Register			x	Per function type.
Cache Line Size Register			x	Does not affect device behavior.
Latency Timer		x		Hard-wired to 00h in PCIe.
Header Type Register		x		
BIST		x		Not supported (00h).
BARs	Memory BAR		x	
	I/O BAR		x	
	MSI-X BAR		x	See Section 9.2.3.3, "MSI-X Capability" .
I/O BAR mapping	IOADDR, IODATA		x	
Subsystem Vendor ID		x		
Subsystem ID		x		
Expansion ROM			x	
Cap_Ptr Register		x		
Interrupt Line Register			x	Just store the register value.
Interrupt Pin Register			x	Separate interrupt number (A-D) per PF.
Min. Grant		x		
Max Latency		x		

9.2.2.1 Vendor ID Register (0x0; RO)

This is a read-only register that has the same value for all PCI functions. It identifies unique Intel products. The value of this field is loaded from the NVM *PCI_VENDOR_ID* field. The default value, if not loaded, is 0x8086.

9.2.2.2 Device ID Register (0x2; RO)

This is a read-only register that identifies individual E610 PCI functions. Both ports have the same default value equals to 0x57AC, and can be autoloading from the NVM during initialization with different values for each port as well as the dummy function (See [Section 4.5](#) for dummy function relevance).

The following table describes the values set in this register from NVM according to the different PHY connections:

Table 9-5. Device ID and Branding String

Device ID	Interface	Branding String
0x57AC	Default Device ID	
0x57AD	Default VF Device ID	
0x57AE	Backplane	
0x57AF	SFP	
0x57B0	10GBASE-T	E610-XT -- where "XT2" indicates 1 port of 10GBASE-T
0x57B0	10GBASE-T	E610-XT2 -- where "XT2" indicates 2 ports of 10GBASE-T
0x57B1	1000BASE-T	E610-AT2 -- where "AT2" indicates 2 ports of 1000BASE-T w/Marvell 55nm PHY
0x57B2	SGMII	

9.2.2.3 Command Register (0x4; RW)

Shaded bits are not used by this implementation and are hard-wired to 0b. Each function has its own Command register. Unless explicitly specified, functionality is the same in both functions.

Bit(s)	Init.	Description
0	0b	I/O Access Enable If I/O BAR is not requested, this bit is hard-wired to zero.
1	0b	Memory Access Enable
2	0b	Enable Mastering Also named Bus Master Enable (BME). - LAN functions RW field. - Dummy function RO as zero field.
3	0b	Special Cycle Monitoring Hard-wired to 0b.
4	0b	MWI Enable Hard-wired to 0b.
5	0b	Palette Snoop Enable Hard-wired to 0b.
6	0b	Parity Error Response
7	0b	Wait Cycle Enable Hard-wired to 0b.
8	0b	SERR# Enable
9	0b	Fast Back-to-Back Enable Hard-wired to 0b.
10	0b	Interrupt Disable When set, devices are prevented from generating legacy interrupt messages.
15:11	0x0	Reserved.

9.2.2.4 Status Register (0x6; RO)

Shaded bits are not used by this implementation and are hard-wired to 0b. Each function has its own Status register. Unless explicitly specified, functionality is the same in both functions.

Bit(s)	Init.	Type	Description
2:0	000b		Reserved.
3	0b	RO	Interrupt Status ¹
4	1b	RO	New Capabilities Indicates that a device implements extended capabilities. The E610 sets this bit and implements a capabilities list to indicate that it supports PCI Power Management, Message Signaled Interrupts (MSI), Enhanced Message Signaled Interrupts (MSI-X), VPD and the PCIe extensions.
5	0b		66 MHz Capable Hard-wired to 0b.
6	0b		Reserved.
7	0b		Fast Back-to-Back Capable Hard-wired to 0b.
8	0b	RW1C	Data Parity Reported
10:9	00b		DEVSEL Timing Hard-wired to 0b.
11	0b	RW1C	Signaled Target Abort
12	0b	RW1C	Received Target Abort
13	0b	RW1C	Received Master Abort
14	0b	RW1C	Signaled System Error
15	0b	RW1C	Detected Parity Error

1. The *Interrupt Status* field is a RO field that indicates that an interrupt message is pending internally to the device.

9.2.2.5 Revision Register (0x8; RO)

The default revision ID of this device is 0x00 for E610 A0 and 0x01 for E610 B0. The value of the rev ID is a logic XOR between the default value and the value in the NVM PCIe Device Revision ID (*PCI_REVID*). Note that LAN 0 and LAN 1 functions have the same revision ID.

9.2.2.6 Class Code Register (0x9; RO)

The class code is a read-only value that identifies the device functionality according to the value of the *Storage Class* bit in the NVM *PCI_CLASS* NVM register.

- Class Code = 0x020000 (Ethernet Adapter) if NVM->*Storage Class* = 0b
- Class Code = 0x010000 (SCSI Storage device) if NVM->*Storage Class* = 1b

In the dummy function the class code equals to 0xFF0000.

9.2.2.7 Cache Line Size Register (0xC; RW)

This field is implemented by PCIe devices as a read/write field for legacy compatibility purposes but has no impact on any PCIe device functionality. The default value is zero.

9.2.2.8 Latency Timer (0xD; RO)

Not used. Hard-wired to 0b.

9.2.2.9 Header Type Register (0xE; RO)

This indicates if a device is single- or multi-function. If a single LAN function is the only active one, this field has a value of 0x00 to indicate a single function device. If other functions are enabled, this field has a value of 0x80 to indicate a multi-function device. Table 9-6 lists the different options to set the header type field.

Table 9-6. Header Type Settings

LAN 0 Enable	LAN 1 Enable	Cross-Mode Enable	Header Type Expected Value
0	0	X	N/A (no function)
1	0	0	0x00
0	1	0	0x80 (dummy exist)
1	1	X	0x80 (dual function)
1	0	1	0x80 (dummy exist)
0	1	1	0x00

9.2.2.10 Subsystem Vendor ID Register (0x2C; RO)

This value can be loaded automatically from the NVM at power up or reset. A value of 0x8086 is the default for this field at power up if the NVM does not respond or is not programmed. All functions are initialized to the same value.

9.2.2.11 Subsystem ID Register (0x2E; RO)

This value can be loaded automatically from the NVM at power up with a default value of 0x0000.

PCI Function	Default Value	NVM Address
LAN Functions	0x0000	0x0B

9.2.2.12 Cap_Ptr Register (0x34; RO)

The *Capabilities Pointer* field (*Cap_Ptr*) is an 8-bit field that provides an offset in the E610's PCI configuration space for the location of the first item in the capabilities linked list. The E610 supports this field and implements a capabilities list. Its value is 0x40, which is the address of the first entry: PCI power management.

9.2.2.13 Interrupt Line Register (0x3C; RW)

Read/write register programmed by software to indicate which of the system interrupt request lines the E610's interrupt pin is bound to. Refer to the PCI definition for more details. Each PCI function has its own register.

9.2.2.14 Interrupt Pin Register (0x3D; RO)

Read-only register. LAN 0 / LAN 1 — A value of 0x1...0x4 indicates that this function implements a legacy interrupt on INTA#...INTD#, respectively. Loaded from the PCI_CNF NVM word per function. For dummy function the returned value is 0x0 (Function uses no legacy interrupt Message).

Note: If only a single device/function of the E610 component is enabled, this value is ignored and the *Interrupt Pin* field of the enabled device reports INTA# usage.

9.2.2.15 Max_Lat and Min_Gnt (0x3E; RO)

Not used. Hard-wired to 0b.

For Dummy functions this register is RO - zero.

9.2.2.16 Memory and IO Base Address Registers (0x10...0x27; RW)

Base Address Registers (BARs) are used to map the E610 register space of the device functions. The E610 has a memory BAR, I/O BAR and MSI-X BAR described in [Table 9-7](#). The BARs location and sizes are described in the [Table 9-7](#) and [Table 9-9](#). The fields within each BAR are then described in [Table 9-9](#).

Table 9-7. E610 Base Address Registers Description — LAN 0 / LAN 1

Mapping Windows	Mapping Description
Memory BAR	The internal registers memories and external Flash device are accessed as direct memory mapped offsets from the BAR. Software can access a DWord or 64 bits. The Flash space in this BAR is enabled by the <i>FL_BAR_SIZE</i> and <i>CSRSize</i> fields in the <i>PCI_LBARCTRL</i> register. Address 0 in the Flash device is mapped to address 1 M in the memory BAR. When the usable Flash size + CSR space are smaller than the memory BAR, accessing addresses above the top of the Flash wraps back to the beginning of the Flash.
I/O BAR	All internal registers and memories can be accessed using I/O operations. There are two 4-byte registers in the I/O mapping window: Addr Reg and Data Reg accessible as DWord entities. The I/O BAR is supported depending on the <i>IO_Sup</i> bit in the NVM at word <i>PCIe Control 3 – Offset 0x07</i> .
MSI-X BAR	The MSI-X vectors and Pending Bit Array (PBA) structures are accessed as direct memory mapped offsets from the MSI-X BAR. Software can access DWord entities.

Table 9-8. E610 Base Address Setting in 64-bit BARs Mode

BAR	Addr	31	5	4	3	2	1	0
0	0x10	Memory CSR + FLASH BAR Low see Table 9-9			0/1	1	0	0
1	0x14	Memory CSR + FLASH BAR High (RW)						
2	0x18	IO BAR (RW — 31:5)		0	0	0	0	1
3	0x1C	Reserved (RO — 0)						
4	0x20	MSI-X BAR Low (RW — 31:14; RO 0b — 13:4)			0/1	1	0	0
5	0x24	MSI-X BAR High (RW)						

Table 9-9. Base Address Registers Fields

Field	Bit(s)	Type	Description
Memory and I/O Space Indication	0	RO	Memory and I/O Space Indication 0b = Indicates memory space. 1b = Indicates I/O.
Memory Type	2:1	RO	Memory Type 00b = <i>Reserved</i> . 10b = 64-bit BAR
Prefetch Memory	3	RO	Prefetch Memory 0b = Non-prefetchable space. 1b = Prefetchable space. This bit should be set only in systems that do not generate prefetchable cycles. This bit is loaded from the <i>PREFBAR</i> bit in the NVM because it is required for 64-bit memory BARs.
Address Space (low register for 64-bit memory BARs)	31:4	RW	Address Space The length of the RW bits and RO 0b bits depend on the mapping window size. Initial value of the RW fields is 0x0. The size of the memory BAR is described in Table 9-10 .
			Mapping Window
			RO bits
			Memory CSR + Flash BAR size depends on <i>PCI_LBARCTRL.FL_BAR_SIZE</i> and <i>PCI_LBARCTRL.CSRSize</i> fields.
			17:4 for 256 KB 18:4 for 512 KB and so on...
			MSI-X space is 16 KB.
			13:4
			I/O space size is 32 bytes.
			4:0

Table 9-10. Usable FLASH Size and CSR Mapping Window Size

FL_BAR_SIZE	CSRSize	Resulted CSR + FLASH BAR Size ^{1 2}	Installed FLASH Device	Usable FLASH Space ¹
000b-100b	X	Reserved		
101b	0	2 MB	2 MB	2 MB minus 1 MB
101b	1	4 MB	2 MB	2 MB
110b	0	4 MB	4 MB	4 MB minus 1 MB
110b	1	8 MB	4 MB	4 MB
111b	0	8 MB	8 MB	8 MB minus 1 MB
111b	1	16 MB	16 MB	16 MB minus 1 MB

1. In the dummy function, the FLASH is unusable and the resulted CSR + FLASH BAR Size is 4KB
2. All values in this table are for active FLASH_EXPOSE flag in the PCI_LBARCTRL register. If the FLASH_EXPOSE flag is inactive, then the Resulted CSR + FLASH BAR Size is 2MB regardless of the FLASH size. If the FLASH_EXPOSE flag is inactive, the FLASH is accessible only via the Expansion ROM BAR.

9.2.2.17 Expansion ROM Base Address Register (0x30; RW)

This register is used to define the address and size information for boot-time access to the Expansion ROM Module in the Flash memory. It is enabled by the PCI_LBARCTRL.EXROM_DIS register field. This register returns a zero value for functions without an expansion ROM window and for dummy functions.

Field	Bit(s)	Init.	Type	Description
En	0	0b	RW	Enable 0b = Disables expansion ROM access. 1b = Enables expansion ROM access.
Reserved	10:1	0x0	R	Reserved. Always read as 0x0. Writes are ignored.
Address	31:11	0x0	RW	Address Read-write bits are hard-wired to 0b and dependent on the memory mapping window size. The LAN Expansion ROM spaces can be either 64 KB or up to 8MB in powers of 2. Mapping window size is set by PCI_LBARCTRL.EXROM_BAR_SIZE field.

9.2.3 PCI Capabilities

The first entry of the PCI capabilities link list is pointed to by the Cap_Ptr register. Table 9-11 lists the capabilities supported by the E610.

Table 9-11. PCI Capabilities List for LAN Functions

Address	Item	Next Pointer
0x40:4F	PCI Power Management	0x50 / 0x70 ¹
0x50:6F	MSI	0xB0
0xB0:CF	MSI-X	0x70
0x70:AF	PCIe Capabilities	0xD0 / 0x00
0xD0:0xDF	VPD Capability	0x00

1. In the dummy function, the power management capability points to the PCIe capabilities.

Table 9-12. PCI Capabilities for Dummy Function

Address	Item	Next Pointer
0x40:47	PCI Power Management	0x70
0x70:AF	PCIe Capabilities	0x00

9.2.3.1 PCI Power Management Capability

All fields are reset at full power up. All fields except *PME_En* and *PME_Status* are reset after exiting from the D3cold state. If AUX power is not supplied, the *PME_En* and *PME_Status* fields also reset after exiting from the D3cold state. Refer to the detailed description for registers loaded from the NVM at initialization.

Table 9-13. PCI Power Management Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x40	Power Management Capabilities		Next Pointer (0x50 / 0xA0)	Capability ID (0x01)
0x44	Data	Bridge Support Extensions	Power Management Control & Status	

Table 9-14 summarizes sharing of the Power Management Capability registers among the different PCI functions.

Table 9-14. Sharing the Power Management Capability Registers

Field	Sub-field	Shared?	Replicated?	Comments
Capability ID		x		
Next Pointer			x	
Power Management Capabilities	PME_Support	x		
	D2_Support	x		
	D1_Support	x		
	AUX Current	x		
	DSI	x		
	PME Clock	x		Hard-wired to 0b.
	Version	x		
Power Management Control / Status	PME_Status		x	
	Data_Scale	x		
	Data_Select		x	
	PME_En		x	
	No_Soft_Reset	x		
	PowerState		x	
Data Register			x	

9.2.3.1.1 Capability ID Register (0x40; RO)

This field equals 0x01 indicating the linked list item as being the PCI Power Management registers.

9.2.3.1.2 Next Pointer Register (0x41; RO)

This field provides an offset to the next capability item in the capability list. This field equals for both LAN ports to 0x50 pointing to the MSI capability. In dummy function, it equals to 0xA0 pointing to the PCIe Capabilities.

9.2.3.1.3 Power Management Capabilities — PMC Register (0x42; RO)

This register describes the device functionality during the power management states as listed in the following table. Note that each device function has its own register.

Bit(s)	Init.	Type	Description
2:0	011b	RO	Version The E610 complies with the PCI PM specification revision 1.2.
3	0b	RO	PME_Clock Disabled. Hard-wired to 0b.
4	0b	RO	Reserved.
5	1b	RO	DSI The E610 requires its device driver to be executed following a transition to the D0 uninitialized state.
8:6	000b	RO	AUX Current Required current defined in the Data register.
9	0b	RO	D1_Support The E610 does not support the D1 state.
10	0b	RO	D2_Support The E610 does not support the D2 state.
15:11	01001b	RO	PME_Support This 5-bit field indicates the power states in which the function can assert PME#. Condition Functionality Values: 01001b = No AUX Pwr - PME at D0 and D3hot 11001b = AUX Pwr - PME at D0, D3hot, and D3cold Note: For dummy function, this field is RO - zero.

9.2.3.1.4 Power Management Control/Status Register — PMCSR (0x44; RW)

This register is used to control and monitor power management events in the device. Note that each device function has its own PMCSR.

Bit(s)	Init.	Type	Description
1:0	00b	RW	PowerState This field is used to set and report the power state of a function as follows: 00b = D0 01b = D1 (cycle ignored if written with this value) 10b = D2 (cycle ignored if written with this value) 11b = D3
2	0b	RO	Reserved for PCIe.
3	1b	RO	No_Soft_Reset This bit is always set to 1b to indicate that the E610 does not perform an internal reset upon transition from D3hot to D0 via software control of the <i>PowerState</i> bits. Configuration context is kept as part of the transition from the D3hot to the D0 state, thus a full re-initialization sequence of the configuration space is not needed to return the E610 to the D0 Initialized state.
7:4	0x0	RO	Reserved.
8	0b at power up	RWS	PME_En If power management is enabled in the NVM. Writing a 1b to this register enables Wake-up.
12:9	0x0	RW	Data_Select This 4-bit field is used to select which data is to be reported through the Data register and <i>Data_Scale</i> field. These bits are writable only when power management is enabled via the NVM.

Bit(s)	Init.	Type	Description
14:13	01b	RO	Data_Scale This field indicates the scaling factor that is used when interpreting the value of the Data register. This field equals 01b (indicating 0.1 watt/units) if the <i>Data_Select</i> field is set to 0, 3, 4, 7, (or 8 for function 0 in multi-function device). Otherwise, it equals 00b.
15	0b at power up	RW1CS	PME_Status This bit is set to 1b when the function detects a wake-up event independent of the state of the <i>PME_En</i> bit. Writing a 1b clears this bit.

9.2.3.1.5 PMCSR_BSE Bridge Support Extensions Register (0x46; RO)

This register is not implemented in the E610; values set to 0x00.

9.2.3.1.6 Data Register (0x47; RO)

This optional register is used to report power consumption and heat dissipation. The reported register is controlled by the *Data_Select* field in the PMCSR; the power scale is reported in the *Data_Scale* field in the PMCSR. The data for this field is loaded from the NVM via the PCI_PWRDATA register if power management is enabled in the NVM or with a default value of 0x00. The values for the E610's functions are as follows (the relevant column is selected based on the value of the *Data_Select* field):

Function	D0 (Consume/ Dissipate)	D3 (Consume/ Dissipate)	Common	Data_Scale
Data_Select	(0x0/0x4)	(0x3/0x7)	(0x8)	
0	PCI_PWRDATA.D0_POWER	PCI_PWRDATA.D3_POWER	Function zero of a Multi-function device: PCI_PWRDATA.COMM_POWER Single-function device: 0x00	01b
1	PCI_PWRDATA.D0_POWER	PCI_PWRDATA.D3_POWER	0x00	01b

Note: For other *Data_Select* values the Data register output is reserved (0b).

9.2.3.2 MSI Capability

Note: This capability is not available for dummy functions.

This structure is required for PCIe devices.

Table 9-15. MSI Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x50	Message Control (0x0080)		Next Pointer (0x70)	Capability ID (0x05)
0x54	Message Address			
0x58	Message Upper Address			
0x5C	Reserved		Message Data	
0x60	Mask Bits			
0x64	Pending Bits			

Table 9-16 summarizes configuration sharing of the MSI Capability registers among the different PCI functions.

Table 9-16. Configuration Sharing of the MSI Capability

Field	Sub-field	Shared?	Replicated?	Comments
Capability ID		x		
Next Pointer			x	
Message Control	MSI Enable		x	
	Multiple Messages Capable	x		
	Multiple Message Enable		x	
	64-bit Capable	x		
	MSI per-vector masking	x		
Message Address Low			x	
Message Address High			x	
Message Data			x	
Mask Bits			x	
Pending Bits			x	

9.2.3.2.1 Capability ID Register (0x50; RO)

This field equals 0x05 indicating that the linked list item as being the MSI registers.

9.2.3.2.2 Next Pointer Register (0x51; RO)

This field provides an offset to the next capability item in the capability list. Its value of 0xB0 and points to MSI-X capability.

9.2.3.2.3 Message Control Register (0x52; RW)

Note: There is a dedicated register (per PCI function) to separately enable its MSI.

Bit(s)	Init.	Type	Description
0	0b	RW	MSI Enable 1b = Message Signaled Interrupts — The E610 generates an MSI for interrupt assertion instead of INTx signaling.
3:1	000b	RO	Multiple Messages Capable The E610 indicates a single requested message per function.
6:4	000b	RW	Multiple Message Enable Since the E610 requests a single vector in the <i>Multiple Message Capable</i> field, software is expected to write 000b to this field.
7	1b	RO	64-bit Capable A value of 1b indicates that the E610 is capable of generating 64-bit message addresses.
8	1b ¹	RO	MSI per-vector masking A value of 0b indicates that the E610 is not capable of per-vector masking. A value of 1b indicates that the E610 is capable of per-vector masking.
15:9	0x0	RO	Reserved. Reads as 0x0.

1. The value is loaded from the MSI Mask bit in the NVM.

9.2.3.2.4 Message Address Low Register (0x54; RW)

Written by the system to indicate the lower 32 bits of the address to use for the MSI memory write transaction. The lower two bits always return 0b regardless of the write operation.

9.2.3.2.5 Message Address High Register (0x58; RW)

Written by the system to indicate the upper 32 bits of the address to use for the MSI memory write transaction.

9.2.3.2.6 Message Data Register (0x5C; RW)

Written by the system to indicate the lower 16 bits of the data written in the MSI memory write DWord transaction. The upper 16 bits of the transaction are written as 0b.

9.2.3.2.7 Mask Bits Register (0x60; RW)

The Mask Bits and Pending Bits registers enable software to disable or defer message sending on a per-vector basis. As the E610 supports only one message, only bit 0 of these registers are implemented.

Bit(s)	Init.	Type	Description
0	0b	RW	MSI Vector 0 Mask If set, the E610 is prohibited from sending MSI messages.
31:1	0x0	RO	Reserved.

9.2.3.2.8 Pending Bits Register (0x64; RW)

Bit(s)	Init.	Type	Description
0	0b	RO	MSI Message If set, the E610 has a pending MSI message.
31:1	0x0	RO	Reserved.

9.2.3.3 MSI-X Capability

Note: This capability is not available for dummy functions.

More than one MSI-X capability structure per function is prohibited while a function is permitted to have both an MSI and an MSI-X capability structure.

In contrast to the MSI capability structure, which directly contains all of the control/status information for the function's vectors, the MSI-X capability structure instead points to an MSI-X table structure and an MSI-X Pending Bit Array (PBA) structure, each residing in memory space.

Each structure is mapped by a BAR belonging to the function that begins at 0x10 in the configuration space. A BAR Indicator Register (BIR) indicates which BAR and a QWord-aligned offset indicates where the structure begins relative to the base address associated with the BAR. The BAR is 64-bit, but must map to the memory space. A function is permitted to map both structures with the same BAR or map each structure with a different BAR.

The MSI-X table structure (Section 9.2.3.4) typically contains multiple entries, each consisting of several fields: *Message Address*, *Message Upper Address*, *Message Data*, and *Vector Control*. Each entry is capable of specifying a unique vector.

The PBA structure [MSI-X Table Offset Register (0xB4; RW)] contains the function's pending bits, one per table entry, organized as a packed array of bits within QWords. The last QWord is not necessarily fully populated.

To request service using a given MSI-X table entry, a function performs a DWord memory write transaction using:

- The contents of the *Message Data* field entry for data.
- The contents of the *Message Upper Address* field for the upper 32 bits of the address.
- The contents of the *Message Address* field entry for the lower 32 bits of the address.

A memory read transaction from the address targeted by the MSI-X message produces undefined results.

The MSI-X table and MSI-X PBA are permitted to co-reside within a naturally aligned 4 KB address range, though they must not overlap with each other.

MSI-X table entries and *Pending* bits are each numbered 0 through N-1, where N-1 is indicated by the *Table Size* field in the MSI-X Message Control register. For a given arbitrary MSI-X table entry K, its starting address can be calculated with the formula:

Entry starting address = Table base + K*16

For the associated *Pending* bit K, its address for QWord access and bit number within that QWord can be calculated with the formulas:

QWord address = PBA base + (K div 64)*8

QWord bit# = K mod 64

Software that chooses to read *Pending* bit K with DWord accesses can use these formulas:

DWord address = PBA base + (K div 32)*4

DWord bit# = K mod 32

Table 9-17. MSI-X Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0xB0	Message Control (0x00090)		Next Pointer	Capability ID (0x11)
0xB4	Table Offset			
0xB8	PBA Offset			

Table 9-18 summarizes configuration sharing of the MSI-X Capability registers among the different PCI functions.

Table 9-18. Configuration Sharing of the MSI-X Capability

Field	Sub-field	Shared?	Replicated?	Comments
Capability ID		x		
Next Pointer			x	
Message Control	Table Size	x		
Function Mask			x	
MSI-X Enable			x	
MSI-X Table Offset	Table BIR		x	
	Table Offset		x	
MSI-X Pending Bit Array	PBA BIR		x	
	PBA Offset		x	
MSI-X Table			x	
MSI-X PBA Structure			x	

9.2.3.3.1 Capability ID Register (0xB0; RO)

This field equals 0x11 indicating that the linked list item as being the MSI-X registers.

9.2.3.3.2 Next Pointer Register (0xB1; RO)

This field provides an offset to the next capability item in the capability list. Its value of 0x70 points to PCIe capability.

9.2.3.3.3 Message Control Register (0xB2; RW)

Note: There is a dedicated register (per PCI function).

Bit(s)	Init.	Type	Description
10:0	0x3F	RO	Table Size System software reads this field to determine the MSI-X Table Size N, which is encoded as N-1. The E610 supports up to 64 different interrupt vectors per function. This field is loaded from the PCI_CNF2.MSI_X_PF_N register field.
13:11	000b	RO	Reserved. Always returns 000b on a read. A write operation has no effect.
14	0b	RW	Function Mask 0b = Each vector's <i>Mask</i> bit determines whether the vector is masked or not. 1b = All of the vectors associated with the function are masked, regardless of their per-vector <i>Mask</i> bit states. Setting or clearing the MSI-X <i>Function Mask</i> bit has no effect on the state of the per-vector <i>Mask</i> bits.
15	0b	RW	MSI-X Enable 0b = The function is prohibited from using MSI-X to request service. 1b = If 1b and the MSI <i>Enable</i> bit in the MSI Message Control register is 0b, the function is permitted to use MSI-X to request service and is prohibited from using its INTx# pin. System configuration software sets this bit to enable MSI-X. A device driver is prohibited from writing this bit to mask a function's service request.

9.2.3.3.4 MSI-X Table Offset Register (0xB4; RW)

Bit(s)	Init.	Type	Description
2:0	0x4	RO	Table BIR Indicates which one of a function's BARs, beginning at 0x10 in the configuration space, is used to map the function's MSI-X table into the memory space. while BIR values: 0...5 correspond to BARs 0x10...0x 24, respectively.
31:3	0x0	RO	Table Offset Used as an offset from the address contained in one of the function's BARs to point to the base of the MSI-X table. The lower three <i>Table BIR</i> bits are masked off (set to 0b) by software to form a 32-bit QWord-aligned offset. This field is read only.

9.2.3.3.5 MSI-X Pending Bit Array — PBA Offset (0xB8; RW)

Bit(s)	Init.	Type	Description
2:0	0x4	RO	PBA BIR Indicates which one of a function's BARs, beginning at 0x10 in the configuration space, is used to map the function's MSI-X PBA into the memory space. while BIR values: 0...5 correspond to BARs 0x10...0x 24, respectively.
31:3	0x0400	RO	PBA Offset Used as an offset from the address contained in one of the functions BARs to point to the base of the MSI-X PBA. The lower three <i>PBA BIR</i> bits are masked off (set to 0b) by software to form a 32-bit QWord-aligned offset. This field is read only.

9.2.3.4 MSI-X Table Structure

DWord3 — MSIXTVCTRL	DWord2 — MSIXMSG	DWord1 — MSIXTUADD	DWord0 — MSIXTADD	Entry Number	BAR 3 — Offset
Vector Control	Msg Data	Msg Upper Address	Msg Lower Address	0	Base (0x0000)
Vector Control	Msg Data	Msg Upper Address	Msg Lower Address	1	Base + 1*16
Vector Control	Msg Data	Msg Upper Address	Msg Lower Address	2	Base + 2*16
...	...	...	...	...	
Vector Control	Msg Data	Msg Upper Address	Msg Lower Address	63	Base + 63*16
Vector Control	Msg Data	Msg Upper Address	Msg Lower Address	64	Base + 64*16
...	...	...	...	...	
Vector Control	Msg Data	Msg Upper Address	Msg Lower Address	255	Base + 255*16

Note: All MSI-X vectors > MSI-X 63, are usable only by the Virtual Functions (VFs) in IOV mode. These vectors are not exposed to the operating system by the *Table Size* field in the *MSI-X Message Control* word.

See [Section 8.2.2.7](#) for details of the MSI-X registers in BAR 4 of the PF.

9.2.3.5 VPD Registers

Note: This capability is not available for dummy functions.

The E610 supports access to a VPD structure stored in the NVM using the following set of registers.

Initial values of the configuration registers are marked in parenthesis.

Note: The VPD structure is available through both ports functions. As the interface is common to the two functions, accessing the VPD structure of one function while an access to the NVM is in process on the other function can yield to unexpected results.

Table 9-19. VPD Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0xD0	VPD Address		Next Pointer (0x00)	Capability ID (0x03)
0xD4	VPD Data			

Table 9-20 summarizes configuration sharing of the VPD Capability registers among the different PCI functions.

Table 9-20. Configuration Sharing of the VPD Capability

Field	Shared?	Replicated?	Comments
Capability ID	X		
Next Pointer	X		
VPD Address/F	X		
VPD Data	X		

9.2.3.5.1 Capability ID Register (0xD0; RO)

This field equals 0x3 indicating the linked list item as being the VPD registers.

9.2.3.5.2 Next Pointer Register (0xD1; RO)

Offset to the next capability item in the capability list. A 0x00 value indicates that it is the last item in the capability-linked list.

9.2.3.5.3 VPD Address Register (0xD2; RW)

Word-aligned byte address of the VPD area in the NVM to be accessed. The register is read/write, and the initial value at power-up is indeterminate.

Bit(s)	Init.	Type	Description
14:0	X	RW	Address DWord-aligned byte address of the VPD area in the NVM to be accessed. The register is read/write, and the initial value at power-up is indeterminate. The two LSBs are RO as zero.
15	0b	RW	F A flag used to indicate when the transfer of data between the VPD Data register and the storage component completes. The Flag register is written when the VPD Address register is written. 0b = Read — Set by hardware when data is valid. 1b = Write — Cleared by hardware when data is written to the NVM. The VPD address and data should not be modified before the action is done.

9.2.3.5.4 VPD Data Register (0xD4; RW)

VPD read/write data.

Bit(s)	Init.	Type	Description
31:0	X	RW	VPD Data VPD data can be read or written through this register. The LS byte of this register (at offset 4 in this capability structure) corresponds to the byte of VPD at the address specified by the VPD Address register. The data read from or written to this register uses the normal PCI byte transfer capabilities. Four bytes are always transferred between this register and the VPD storage component. Reading or writing data outside of the VPD space in the storage component is not allowed. In a write access, the data should be set before the address and the flag is set.

9.2.3.6 PCIe Capability

The E610 implements the PCIe capability structure linked to the legacy PCI capability list for endpoint devices as follows:

Table 9-21. PCIe Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x70	PCI Express Capability Register (0x0002)		Next Pointer (0xE0/0x00)	Capability ID (0x10)
0x74	Device Capability			
0x78	Device Status		Device Control	
0x7C	Link Capability			
0x80	Link Status		Link Control	
0x84	Reserved			
0x88	Reserved		Reserved	
0x8C	Reserved			
0x90	Reserved		Reserved	
0x94	Device Capability 2			
0x98	Reserved		Device Control 2	
0x9C	Reserved			
0xA0	Link Status 2		Link Control 2	
0xA4	Reserved			
0xA8	Reserved		Reserved	

Table 9-22 summarizes configuration sharing of the PCIe Capability registers among the different PCI functions.

Table 9-22. Configuration Sharing of the PCIe Capability

Field	Sub-field	Shared?	Replicated?	Comments
Capability ID		x		
Next Pointer			x	
PCIe Capabilities		x		
Device Capabilities	Max Payload Size Supported	x		
	Phantom Functions Supported	x		Not supported.
	Extended Tag Field Supported	x		
	Endpoint L0s Acceptable Latency	x		
	Endpoint L1 Acceptable Latency	x		
	Function Level Reset Capability	x		
Device Control	Correctable Error Reporting Enable		x	
	Non-Fatal Error Reporting Enable		x	
	Fatal Error Reporting Enable		x	
	Unsupported Request Reporting Enable		x	
	Enable Relaxed Ordering		x	
	Max Payload Size		x	Use minimum of all configured values. In ARI device, use value in Function 0.
	Extended Tag field Enable		x	
	Auxiliary Power PM Enable		x	Same policy for all PFs (Logical OR of the PF's bits)
	Enable No Snoop		x	
	Max Read Request Size		x	Use minimum of all configured values.
	Initiate Function Level Reset		x	
Device Status	Correctable Detected		x	
	Non-Fatal Error Detected		x	
	Fatal Error Detected		x	
	Unsupported Request Detected	x		
	Aux Power Detected	x		
	Transactions Pending		x	
Link Capabilities	Supported Link Speeds	x		
	Max Link Width	x		
	Active State Link PM Support	x		
	L0s Exit Latency	x		
	L1 Exit Latency	x		
	Clock Power Management	x		
	Port Number	x		

Table 9-22. Configuration Sharing of the PCIe Capability [continued]

Field	Sub-field	Shared?	Replicated?	Comments
Link Control	Active State Link PM Control		x	Same policy for all PFs (Logical AND of the PF's bits). In ARI device, use value in Function 0.
	Read Completion Boundary (RCB)		x	
	Common Clock Configuration		x	Same policy for all PFs (Logical AND of the PF's bits). In ARI device, use value in Function 0.
	Extended Sync		x	Same policy for all PFs (Logical OR of the PF's bits).
Link Status	Current Link Speed	x		
	Negotiated Link Width	x		
	Slot Clock Configuration	x		
Device Capabilities 2	Completion Timeout Ranges Supported	x		
	Completion Timeout Disable Supported	x		
	LTR Mechanism Supported	x		
	TPH Completer Supported	x		
	Extended Fmt Field Supported	x		
	OBFF Supported	x		
Device Control 2	Completion Timeout Value		x	Completion timeout decision per PF or use the largest configured value among PFs.
	Completion Timeout Disable		x	Completion timeout mechanism enabled per PF.
	IDO Request Enable		x	
	IDO Completion Enable		x	
	LTR Mechanism Enable		x	PF0 only. RsvdP on other functions.
	OBFF Enable		x	PF0 only. RsvdP on other functions.
Link Capabilities 2		x		
Link Control 2		x		PF0 only. RsvdP on other functions.
Link Status 2		x		

9.2.3.6.1 Capability ID Register (0x70; RO)

This field equals 0x10 indicating that the linked list item as being the PCIe Capabilities registers.

9.2.3.6.2 Next Pointer Register (0x71; RO)

Offset to the next capability item in the capability list. Its value of 0xD0 points to the VPD structure. If VPD is disabled or for a dummy function, a value of 0x00 value indicates that it is the last item in the capability-linked list.

9.2.3.6.3 PCIe Capabilities Register (0x72; RO)

The PCIe Capabilities register identifies PCIe device type and associated capabilities. This is a read-only register identical to all functions.

Bit(s)	Init.	Type	Description
3:0	0x2	RO	Capability Version Indicates the PCIe capability structure version. The E610 supports PCIe version 2 (also loaded from the PCI_CAPSUP.PCI_VER bit in the NVM).
7:4	0x0	RO	Device/Port Type Indicates the type of PCIe functions. All functions are native PCI functions with a value of 0x0.
8	0b	RO	Slot Implemented The E610 does not implement slot options. Therefore, this field is hard-wired to 0b.
13:9	0x0	RO	Interrupt Message Number The E610 does not implement multiple MSI per function. As a result, this field is hard-wired to 0x0.
15:14	00b	RO	Reserved.

9.2.3.6.4 Device Capabilities Register (0x74; RO)

This register identifies the PCIe device specific capabilities. It is a read-only register with the same value for the two LAN functions and for all other functions.

Bit(s)	Init.	Type	Description
2:0	010b	RO	Max Payload Size Supported This field indicates the maximum payload that The E610 can support for TLPs. It is loaded from the NVM with a default value of 512 bytes.
4:3	00b	RO	Phantom Function Supported Not supported by the E610.
5	1b	RO	Extended Tag Field Supported Maximum supported size of the <i>Tag</i> field. The E610 supports a 8-bit <i>Tag</i> field for all functions.
8:6	011b	RO	Endpoint L0s Acceptable Latency This field indicates the acceptable latency that the E610 can withstand due to the transition from L0s state to the L0 state. All functions share the same value loaded from the NVM PCIe Init Configuration 1 bits [8:6]. A value of 011b equals 512 ns.
11:9	110b	RO	Endpoint L1 Acceptable Latency This field indicates the acceptable latency that the E610 can withstand due to the transition from L1 state to the L0 state. A value of 110b equals 32 μ s to 64 μ s. All functions share the same value loaded from the NVM.
12	0b	RO	Attention Button Present Hard-wired in the E610 to 0b for all functions.
13	0b	RO	Attention Indicator Present Hard-wired in the E610 to 0b for all functions.
14	0b	RO	Power Indicator Present Hard-wired in the E610 to 0b for all functions.
15	1b	RO	Role Based Error Reporting Hard-wired in the E610 to 1b for all functions.
17:16	000b	RO	Reserved.
25:18	0x0	RO	Slot Power Limit Value Used in upstream ports only. Hard-wired in the E610 to 0x0 for all functions.
27:26	00b	RO	Slot Power Limit Scale Used in upstream ports only. Hard-wired in the E610 to 0b for all functions.

Bit(s)	Init.	Type	Description
28	1b	RO	Function Level Reset Capability A value of 1b indicates the Function supports the optional Function Level Reset (FLR) mechanism.
31:29	000b	RO	Reserved.

9.2.3.6.5 Device Control Register (0x78; RW)

This register controls the PCIe specific parameters. Note that there is a dedicated register per each function.

Bit(s)	Init.	Type	Description
0	0b	RW	Correctable Error Reporting Enable Enable error report.
1	0b	RW	Non-Fatal Error Reporting Enable Enable error report.
2	0b	RW	Fatal Error Reporting Enable Enable error report.
3	0b	RW	Unsupported Request Reporting Enable Enable error report.
4	1b	RW	Enable Relaxed Ordering If this bit is set, the E610 is permitted to set the <i>Relaxed Ordering</i> bit in the <i>Attribute</i> field of write transactions that do not need strong ordering. Refer to the CTRL_EXT register bit <i>RO_DIS</i> for more details.
7:5	000b (128 bytes)	RW	Max Payload Size This field sets the maximum TLP payload size for the E610 functions. As a receiver, the E610 must handle TLPs as large as the set value. As a transmitter, the E610 must not generate TLPs exceeding the set value. The <i>Max Payload Size</i> field supported in the Device Capabilities register indicates permissible values that can be programmed. In ARI device, <i>Max Payload Size</i> is determined solely by the field in function 0 (even when it is a dummy function) while it is meaningless in the other function(s).
8	1b	RW	Extended Tag field Enable The E610 uses 8-bit tags when this bit is set and a 5-bit tag when disabled.
9	0b	RW	Phantom Functions Enable Not implemented in the E610.
10	0b	RWS	Auxiliary Power PM Enable When set, enables the E610 to draw AUX power independent of PME AUX power. The E610 is a multi-function device, therefore allowed to draw AUX power if at least one of the functions has this bit set.
11	0b	RW	Enable No Snoop No-snoop is not used by the E610.
14:12	010b	RW	Max Read Request Size This field sets maximum read request size for the E610 as a requester. 000b = 128 bytes 001b = 256 bytes 010b = 512 bytes 011b = 1024 bytes 100b = 2048 bytes 101b = 4096 bytes (Not supported by the E610) 110b = Reserved 111b = Reserved
15	0b	RW	Initiate FLR A write of 1b initiates FLR to the function. The value read by software from this bit is always 0b.

9.2.3.6.6 Device Status Register (0x7A; RW1C)

This register provides information about PCIe device specific parameters. Note that there is a dedicated register per each function.

Bit(s)	Init.	Type	Description
0	0b	RW1C	Correctable Detected Indicates status of correctable error detection.
1	0b	RW1C	Non-Fatal Error Detected Indicates status of non-fatal error detection.
2	0b	RW1C	Fatal Error Detected Indicates status of fatal error detection.
3	0b	RW1C	Unsupported Request Detected Indicates that the E610 received an unsupported request. This field is identical in all functions. The E610 cannot distinguish which function causes the error.
4	0b	RO	Aux Power Detected If Aux Power is detected, this field is set to 1b. It is a strapping signal from the periphery and is identical for all functions. Resets on LAN Power Good and PE_RST_N only.
5	0b	RO	Transaction Pending Indicates whether the E610 has ANY transactions pending. (transactions include completions for any outstanding non-posted request for all used traffic classes).
15:6	0x0	RO	Reserved.

9.2.3.6.7 Link Capabilities Register (0x7C; RO)

This register identifies PCIe link-specific capabilities. This is a read-only register identical to all functions.

Bit(s)	Init.	Type	Description
3:0	0x4	RO	Supported Max Link Speed This field indicates the supported Link speed(s) of the associated link port. Defined encodings are: 0001b = 2.5 GT/s link speed supported. 0010b = 5 GT/s and 2.5 GT/s link speeds supported. 0011b = 8 GT/s and 5 GT/s and 2.5 GT/s link speeds supported 0100b = 16GT/s and 8 GT/s and 5 GT/s and 2.5 GT/s link speeds supported
9:4		RO	Max Link Width Indicates the maximum link width. The E610 supports a x1, x2 and x4-link width. This field is loaded from the NVM and is reflected in the PCI_LINKCAP register. Defined encoding: 000000b = Reserved 000001b = x1 000010b = x2 000100b = x4
11:10	10b	RO	Active State Link PM Support Indicates the level of the active state of power management supported in the E610. Defined encodings are: 00b = No ASPM support. 01b = L0s Entry supported. 10b = L1 supported. 11b = L0s and L1 supported. All functions share the same value loaded from the NVM.

Bit(s)	Init.	Type	Description
14:12	101b (1 μ s – 2 μ s)	RO	L0s Exit Latency Indicates the exit latency from L0s to L0 state. 000b = Less than 64 ns. 001b = 64 ns – 128 ns 010b = 128 ns – 256 ns 011b = 256 ns – 512 ns 100b = 512 ns – 1 μ s 101b = 1 μ s – 2 μ s 110b = 2 μ s – 4 μ s 111b = Reserved. All functions share the same value loaded from the NVM.
17:15	100b (8 μ s – 16 μ s)	RO	L1 Exit Latency Indicates the exit latency from L1 to L0 state. 000b = Less than 1 μ s 001b = 1 μ s – 2 μ s 010b = 2 μ s – 4 μ s 011b = 4 μ s – 8 μ s 100b = 8 μ s – 16 μ s 101b = 16 μ s – 32 μ s 110b = 32 μ s – 64 μ s 111b = L1 transition not supported. All functions share the same value loaded from the NVM.
18	0	RO	Clock Power Management
19	0	RO	Surprise Down Error Reporting Capable Hard-wired to 0b.
20	0	RO	Data Link Layer Link Active Reporting Capable
21	0	RO	Link Bandwidth Notification Capability Hard-wired to 0b.
22	1b	RO	ASPM Optional Compliance This bit must be set to 1b. Components that were implemented according to an earlier PCIe specification version has this bit set to 0b. Software is permitted to use the value of this bit to help determine whether to enable ASPM or run ASPM compliance tests.
23	0b	RO	Reserved.
31:24	0x0	HwInit	Port Number The PCIe port number for the given PCIe link. This field is set in the link training phase.

9.2.3.6.8 Link Control Register (0x80; RO)

This register controls PCIe link specific parameters. There is a dedicated register per each function.

Bit(s)	Init.	Type	Description
1:0	00b	RW	Active State Link PM Control This field controls the active state PM supported on the link. Link PM functionality is determined by the lowest common denominator of all functions. For non-ARI device, only capabilities enabled in all functions are enabled for the component as a whole. When ARI support is exposed, ASPM control is determined solely by the setting in Function 0 (even when it is a dummy function), regardless of Function 0's D-state. The settings in the other functions always return whatever value software programmed for each, but otherwise are ignored by the E610. Defined encodings are: 00b = PM Disabled. 01b = L0s Entry Supported. 10b = L1 Entry Enabled. 11b = L0s and L1 Supported. In ARI device, the ASPM is determined solely by the field in function 0 (even when it is a dummy function) while it is meaningless in the other function(s).
2	0b	RO	Reserved.
3	0b	RO	Read Completion Boundary
4	0b	RO	Link Disable Reserved for endpoint devices. Hard-wired to 0b.
5	0b	RO	Retrain Clock Not applicable for endpoint devices. Hard-wired to 0b.
6	0b	RW	Common Clock Configuration When set, indicates that the E610 and the component at the other end of the link are operating with a common reference clock. A value of 0b indicates that they are operating with an asynchronous clock. In ARI device, the common clock configuration is determined solely by the field in function 0 (even when it is a dummy function) while it is meaningless in the other function(s).
7	0b	RW	Extended Sync When set, this bit forces an extended Tx of the FTS ordered set in FTS and an extra TS1 at the exit from L0s prior to entering L0.
8	0b	RO	Enable Clock Power Management Not supported in the E610. Hard-wired to 0.
9	0b	RW	Reserved. Returns the value that was written.
10	0b	RO	Link Bandwidth Management Interrupt Enable Not supported in the E610. Hard-wired to 0.
11	0b	RO	Link Autonomous Bandwidth Interrupt Enable Not supported in the E610. Hard-wired to 0.
15:12	0x0	RO	Reserved.

9.2.3.6.9 Link Status Register (0x82; RO)

This register provides information about PCIe Link specific parameters. This is a read only register identical to all functions.

Bit(s)	Init.	Type	Description
3:0	X	RO	Current Link Speed This field indicates the negotiated link speed of the given PCIe link. Defined encodings are: 0001b = 2.5 GT/s PCIe link. 0010b = 5 GT/s PCIe link. 0011b = 8 GT/s PCIe link. 0100b = 16 GT/s PCIe link All other encodings are reserved. The value in this field is undefined when the Link is not up.
9:4	X	RO	Negotiated Link Width Indicates the negotiated width of the link. Relevant encodings for the E610 are: 000001b = x1 000010b = x2 000100b = x4 The value in this field is undefined when the Link is not up.
10	0b	RO	Undefined.
11	0b	RO	Link Training Indicates that link training is in progress. This field is not applicable and is reserved for endpoint devices, and is hard-wired to 0b.
12	1b	HwInit	Slot Clock Configuration When set, indicates that the E610 uses the physical reference clock that the platform provides at the connector. This bit must be cleared if the E610 uses an independent clock. The <i>Slot Clock Configuration</i> bit is loaded from the <i>Slot_Clock_Cfg</i> NVM bit.
13	0b	RO	Data Link Layer Link Active Not supported in the E610. Hard-wired to 0b.
14	0b	RO	Link Bandwidth Management Status Not supported in the E610. Hard-wire to 0b.
15	0b	RO	Link Autonomous Bandwidth Status This bit is not applicable and is reserved for endpoints.

The following registers are supported only if the capability version is two and above.

9.2.3.6.10 Device Capability 2 Register (0x94; RO)

This register identifies the PCIe device-specific capabilities. It is a read-only register with the same value for both LAN functions.

Bit(s)	Init.	Type	Description
3:0	0xF	RO	Completion Timeout Ranges Supported This field indicates the E610's support for the optional completion timeout programmability mechanism. Four time value ranges are defined: - Range A: 50 μs to 10 ms - Range B: 10 ms to 250 ms - Range C: 250 ms to 4 s - Range D: 4 s to 64 s Bits are set according to the following values to show the timeout value ranges that the E610 supports. 0000b = Completion timeout programming not supported. The E610 must implement a timeout value in the range of 50 μs to 50 ms. 0001b = Range A. 0010b = Range B. 0011b = Ranges A and B. 0110b = Ranges B and C. 0111b = Ranges A, B and C. 1110b = Ranges B, C and D. 1111b = Ranges A, B, C and D. All other values are reserved.
4	1b	RO	Completion Timeout Disable Supported A value of 1b indicates support for the completion timeout disable mechanism. Note: For dummy functionality, a completion timeout is not relevant as a dummy function because it never sends non-posted requests.
5	0b	RO	ARI Forwarding Supported Applicable only to Switch Downstream. Ports and Root Ports; must be 0b for other function types.
10:6	0x0	RO	Not supported Hard-wired to 0x0.
11	1b	RO	LTR Mechanism Supported A value of 1b indicates support for the optional Latency Tolerance Reporting (LTR) mechanism. For a multi-Function device associated with an Upstream Port, each Function must report the same value for this bit. Note: Value loaded from <i>LTR_EN</i> bit in the NVM
13:12	00b	RO	TPH Completer Supported Value indicates Completer support for TPH or Extended TPH. This capability is not supported.
15:14	0x0	RO	Reserved.
16	0x1	RO	10-Bit Tag Completer Supported If this bit is Set, the Function supports 10-Bit Tag Completer capability; otherwise, the Function does not. Supported by the E610.
17	0x0	RO	10-Bit Tag Requester Supported If this bit is Set, the Function supports 10-Bit Tag Requester capability; otherwise, the Function does not. Not supported by the E610.
19:18	00b	RO	OBFF Supported 00b = OBFF not supported. 01b = OBFF supported using Message signaling only.. 10b = OBFF supported using WAKE# signaling only 11b = OBFF supported using WAKE# and Message signaling.

Bit(s)	Init.	Type	Description
20	0b	RO	Extended Fmt Field Supported 0b = The Function supports a 2-bit definition of the <i>Fmt</i> field. 1b = The Function supports the 3-bit definition of the <i>Fmt</i> field. Not supported by this device
21	0b	RO	End-End TLP Prefix Supported Indicates whether End-End TLP Prefix support is offered by a Function. Not supported by this device.
23:22	00b	RsvdP	Max End-End TLP Prefixes Indicates the maximum number of End-End TLP Prefixes supported by this Function. Reserved for this device.
31:24	0x0	RO	Reserved.

9.2.3.6.11 Device Control 2 Register (0x98; RW)

This register controls the PCIe specific parameters. Note that there is a dedicated register per each function.

Bit(s)	Init.	Type	Description
3:0	0x0	RW	Completion Timeout Value For devices that support completion timeout programmability, this field enables system software to modify the completion timeout value. Defined encodings: 0000b = Default range: 16 ms to 32 ms. Note: It is strongly recommended that the completion timeout mechanism not expire in less than 10 ms. Values available if Range A (50 μ s to 10 ms) programmability range is supported: 0001b = 50 μ s to 100 μ s. 0010b = 1 ms to 2 ms. Values available if Range B (10 ms to 250 ms) programmability range is supported: 0101b = 16 ms to 32 ms. 0110b = 65 ms to 130 ms. Values available if Range C (250 ms to 4 s) programmability range is supported: 1001b = 260 ms to 520 ms. 1010b = 1 s to 2 s. Values available if the Range D (4 s to 64 s) programmability range is supported: 1101b = 4 s to 8 s. 1110b = 17 s to 34 s. Values not defined are reserved. Software is permitted to change the value of this field at any time. For requests already pending when the completion timeout value is changed, hardware is permitted to use either the new or the old value for the outstanding requests and is permitted to base the start time for each request either on when this value was changed or on when each request was issued. Note: For dummy function, this field is RO - zero.
4	0b	RW	Completion Timeout Disable When set to 1b, this bit disables the completion timeout mechanism. Software is permitted to set or clear this bit at any time. When set, the completion timeout detection mechanism is disabled. If there are outstanding requests when the bit is cleared, it is permitted but not required for hardware to apply the completion timeout mechanism to the outstanding requests. If this is done, it is permitted to base the start time for each request on either the time this bit was cleared or the time each request was issued. Note: For dummy function, this field is RO - zero.
5	0b	RO	ARI Forwarding Enable Applicable only to switch devices.
7:6	00b	RO	Not supported. Hard-wired to 00b.

Bit(s)	Init.	Type	Description
8	0b	RW	IDO Request Enable If this bit is Set, the Function is permitted to set the ID-Based Ordering (IDO) bit (Attribute[2]) of Requests it initiates. Default value of this bit is 0b.
9	0b	RW	IDO Completion Enable If this bit is Set, the Function is permitted to set the ID-Based Ordering (IDO) bit (Attribute[2]) of Completions it returns. Default value of this bit is 0b.
10	0b	RW / RsvdP	LTR Mechanism Enable When Set to 1b, this bit enables Upstream Ports to send LTR messages. For a multi-function device, the bit in Function 0 is RW, and only Function 0 controls the component's Link behavior. In all other Functions of that device, this bit is RsvdP. If the <i>LTR_EN</i> bit in the NVM is 0b, this bit is RO with a value of 0b. Default value of this bit is 0b.
12:11	00b	RO	Reserved
11	0x0	RO	Emergency Power Reduction Request Not supported by the E610.
12	0x0	RW	10-Bit Tag Requester Enable
14:13	00b	RsvdP	OBFF Enable 00b = Disabled 01b = Enabled using Message signaling [Variation A] 10b = Enabled using Message signaling [Variation B] 11b = Enabled using WAKE# signaling
15	0b	RsvdP	End-End TLP Prefix Blocking Not applicable to endpoints.

9.2.3.6.12 Link Capabilities 2 Register (0x9C)

Bit(s)	Init.	Type	Description
0	0b	RO	Reserved.
7:1	0x01	RO	Supported Link Speeds Vector This field indicates the supported Link speed(s) of the associated Port. For each bit, a value of 1b indicates that the corresponding Link speed is supported; otherwise, the Link speed is not supported. Bit definitions are: Bit 1 = 2.5 GT/s Bit 2 = 5.0 GT/s Bit 3 = 8.0 GT/s Bit 4 = 16.0 GT/s Bits 7:5 = RsvdP Multi-Function devices associated with the same Upstream Port must report the same value in this field for all Functions. This field is loaded from NVM and is reflected in the PCI_LINKCAP register.
8	0b	RO	Crosslink Supported When set to 1b, this bit indicates that the associated Port supports cross links. It is recommended that this bit be Set in any Port that supports cross-links even though doing so is only required for Ports that also support operating at 8.0 GT/s or higher Link speeds.
31:9	0x0	RO	Reserved.

9.2.3.6.13 Link Control 2 Register (0xA0; RWS)

All RW fields in this register affect the device behavior only through function 0. In function 1, these fields are reserved read as zeros.

Bit(s)	Init.	Type	Description
3:0	0x4 (func 0) 0x0 (else)	RWS (func 0) RsvdP (else)	Target Link Speed This field is used to set the target compliance mode speed when software is using the <i>Enter Compliance</i> bit to force a link into compliance mode. The encoding is the binary value of the bit in the Supported Link Speeds Vector (in the Link Capabilities 2 register) that corresponds to the desired target Link speed. All other encodings are Reserved. For example, 5.0 GT/s corresponds to bit 2 in the Supported Link Speeds Vector, so the encoding for a 5.0 GT/s target Link speed in this field is 0010b If a value is written to this field that does not correspond to a speed included in the <i>Supported Link Speeds</i> field, the result is undefined. The default value of this field is the highest link speed supported by the E610 (as reported in the <i>Supported Link Speeds</i> field of the Link Capabilities register).
4	0b	RWS (func 0) RsvdP (else)	Enter Compliance Software is permitted to force a link to enter compliance mode at the speed indicated in the <i>Target Link Speed</i> field by setting this bit to 1b in both components on a link and then initiating a hot reset on the link. The default value of this field following a fundamental reset is 0b.
5	0b	RWS (func 0) RsvdP (else)	Hardware Autonomous Speed Disable When set to 1b, this bit disables hardware from changing the link speed for reasons other than attempting to correct unreliable link operation by reducing link speed.
6	0b	RO	Selectable De-Emphasis This bit is not applicable and reserved for endpoints.
9:7	000b	RWS (func 0) RsvdP (else)	Transmit Margin This field controls the value of the non de emphasized voltage level at the Transmitter pins. Encodings: 000b = Normal operating range. 001b = 800-1200 mV for full swing and 400-700 mV for half-swing. 010b = (n-1) — Values must be monotonic with a non-zero slope. The value of n must be greater than 3 and less than 7. At least two of these must be below the normal operating range of n: 200-400 mV for full-swing and 100-200 mV for half-swing. 111b = (n) Reserved.
10	0b	RWS (func 0) RsvdP (else)	Enter Modified Compliance When this bit is set to 1b, the device transmits modified compliance pattern if the LTSSM enters Polling.Compliance state. The default value of this bit is 0b.
11	0b	RWS (func 0) RsvdP (else)	Compliance SOS When set to 1b, the LTSSM is required to send SOS periodically in between the (modified) compliance patterns. This bit is applicable when the Link is operating at 2.5 GT/s or 5 GT/s data rates only. The default value of this bit is 0b.

Bit(s)	Init.	Type	Description
15:12	0x0	RWS (func 0) RsvdP (else)	Compliance Preset/De-emphasis For 8.0 GT/s Data Rate: This field sets the Transmitter Preset in Polling.Compliance state if the entry occurred due to the <i>Enter Compliance</i> bit being 1b. For 5.0 GT/s Data Rate: This field sets the de-emphasis level in Polling.Compliance state if the entry occurred due to the <i>Enter Compliance</i> bit being 1b. When the Link is operating at 2.5 GT/s, the setting of this bit field has no effect. Defined Encodings are: 0001b -3.5 dB 0000b -6 dB For a Multi-Function device associated with an Upstream Port, the bit field in Function 0 is of type RWS, and only Function 0 controls the component's Link behavior. In all other Functions of that device, this bit field is of type RsvdP. This bit field is intended for debug, and compliance testing purposes. The default value of this field is 0000b

9.2.3.7 Link Status 2 Register (0xA2; RW)

Bit(s)	Init.	Type	Description
0	0b	RO	Current De-emphasis Level When the link is operating at 5 GT/s speed, this bit reflects the level of de-emphasis. it is undefined when the Link is operating at 2.5 GT/s speed Encodings: 0b = -6 dB 1b = -3.5 dB
1	0b	ROS (Func 0) RsvdZ (Func 1)	Equalization Complete When set to 1b, this bit indicates that the Transmitter Equalization procedure has completed
2	0b	ROS (Func 0) RsvdZ (Func 1)	Equalization Phase 1 Successful When set to 1b, this bit indicates that Phase 1 of the Transmitter Equalization procedure has successfully completed.
3	0b	ROS (Func 0) RsvdZ (Func 1)	Equalization Phase 2 Successful When set to 1b, this bit indicates that Phase 2 of the Transmitter Equalization procedure has successfully completed.
4	0b	ROS (Func 0) RsvdZ (Func 1)	Equalization Phase 3 Successful When set to 1b, this bit indicates that Phase 3 of the Transmitter Equalization procedure has successfully completed.
5	0b	RW1C (Func 0) RsvdZ (Func 1)	Link Equalization Request This bit is Set by hardware to request the Link equalization process to be performed on the Link. This bit is available only in function zero.
15:6	0x0	RsvdZ	Reserved.

9.2.4 PCIe Extended Configuration Space

PCIe configuration space is located in a flat memory-mapped address space. PCIe extends the configuration space beyond the 256 bytes available for PCI to 4096 bytes. The E610 decodes an additional four bits (bits 27:24) to provide the additional configuration space as shown. PCIe reserves the remaining four bits (bits 31:28) for future expansion of the configuration space beyond 4096 bytes.

The configuration address for a PCIe device is computed using a PCI-compatible bus, device, and function numbers as follows:

31	28	27	20	19	15	14	12	11	2	1	0
0000b		Bus #		Device #		Fun #		Register Address (offset)			00b

PCIe extended configuration space is allocated using a linked list of optional or required PCIe extended capabilities following a format resembling PCI capability structures. The first PCIe extended capability is located at offset 0x100 in the device configuration space. The first DWord of the capability structure identifies the capability/version and points to the next capability.

The E610 supports the following PCIe extended capabilities:

Table 9-23. Extended Capabilities List

Capability	Offset	Next Header	Section Number
Advanced Error Reporting	0x100	Any of the below / 0x000	9.2.4.1
Serial Number	0x148	Any of the below / 0x000 ¹	9.2.4.2
Alternative RID Interpretation (ARI)	0x158	Any of the below / 0x000 ¹	9.2.4.3
SR-IOV Support ²	0x1C4	Any of the below / 0x000	9.2.4.4
TPH Requester	0x204	Any of the below / 0x000	9.2.4.5
Access Control Services (ACS) ³	0x290	Any of the below / 0x000	9.2.4.6
Latency Tolerance Reporting (LTR)	0x298	0x168 / 0x000	9.2.4.7
Secondary PCI Express	0x168	0x3D8 / 0x000	9.2.4.8
Data Link Feature	0x3D8	0x188 / 0x000	9.2.4.9
Physical Layer 16.0 GT/s Extended Capability	0x188	0x1AC / 0x000	9.2.4.10
Lane Margining at the Receiver Capability	0x1AC	0x2A0 / 0x000	9.2.4.11
RAS DES Capability (VSEC)	0x2A0	0x3A0 / 0x000	
RAS Data Path Protection Capability (VSEC)	0x3A0	0x580 / 0x000	
Data Object Exchange (DOE) Capability	0x580	0x000	9.2.4.12

1. Depends on NVM settings enabling the ARI/IOV structures.

2. In a dummy function, the IOV and TPH structures are not exposed.

3. When in Single function mode (function 1 is disabled), the ACS capability is not exposed.

9.2.4.1 Advanced Error Reporting Capability (AER)

The PCIe advanced error reporting capability is an optional extended capability to support advanced error reporting. The tables that follow list the PCIe advanced error reporting extended capability structure for PCIe devices.

Table 9-24. AER Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x100	Next Capability offset		Version (0x1)	AER Capability ID (0x0001)
0x104	Uncorrectable Error Status			
0x108	Uncorrectable Error Mask			
0x10C	Uncorrectable Error Severity			
0x110	Correctable Error Status			
0x114	Correctable Error Mask			
0x118	Advanced Error Capabilities and Control Register			
0x11C... 0x128	Header Log			

Table 9-25 summarizes configuration sharing of the AER Capability registers among the different PCI functions.

Table 9-25. Configuration Sharing of the AER Capability

Field	Sub-field	Shared?	Replicated?	Comments
Enhanced Capability Header Register	Extended Capability ID	x		
	Capability Version	x		
	Next Capability Offset		x	
Uncorrectable Error Status			x	
Uncorrectable Error Mask			x	
Uncorrectable Error Severity			x	
Correctable Error Status			x	
Correctable Error Mask			x	
Advanced Error Capabilities and Control	First Error Pointer		x	
	ECRC Generation Capable	x		
	ECRC Generation Enable		x	ECRC insertion is per PF.
	ECRC Check Capable	x		
	ECRC Check Enable		x	See Section 3.1.2.8 .
Header Log			x	

9.2.4.1.1 Advanced Error Reporting Enhanced Capability Header Register (0x100; RO)

Bit(s)	Init.	Type	Description
15:0	0x1	RO	Extended Capability ID PCIe extended capability ID indicating advanced error reporting capability.
19:16	0x2	RO	Version Number PCIe advanced error reporting extended capability version number.
31:20	See description	RO	Next Capability Offset Next PCIe extended capability offset. See Table 9-23 and Table 9-12 for possible values of the next capability offset.

9.2.4.1.2 Uncorrectable Error Status Register (0x104; RW1CS)

The Uncorrectable Error Status register reports error status of individual uncorrectable error sources on a PCIe device. An individual error status bit that is set to 1b indicates that a particular error occurred; software can clear an error status by writing a 1b to the respective bit. Register is cleared by LAN_PWR_GOOD.

Bit(s)	Init.	Type	Description
0	0b	RO	Reserved.
3:1	000b	RsvdP	Reserved.
4	0b	RW1CS	Data Link Protocol Error Status
5	0b	RO	Reserved.
11:6	0x0	RsvdP	Reserved.
12	0b	RW1CS	Poisoned TLP Status
13	0b	RW1CS	Flow Control Protocol Error Status
14	0b	RW1CS	Completion Timeout Status
15	0b	RW1CS	Completer Abort Status
16	0b	RW1CS	Unexpected Completion Status
17	0b	RW1CS	Receiver Overflow Status
18	0b	RW1CS	Malformed TLP Status.
19	0b	RW1CS	ECRC Error Status
20	0b	RW1CS	Unsupported Request Error Status
21	0b	RO	ACS Violation Status Not supported. Hard-wired to 0b.
25:22	0x0	RO	Not Supported.
31:26	0x0	RsvdP	Reserved.

9.2.4.1.3 Uncorrectable Error Mask Register (0x108; RWS)

The Uncorrectable Error Mask register controls reporting of individual uncorrectable errors by device to the host bridge via a PCIe error message. A masked error (respective bit set in mask register) is not reported to the host bridge by an individual device. Note that there is a mask bit per bit of the Uncorrectable Error Status register.

Bit(s)	Init.	Type	Description
0	0b	RO	Reserved.
3:1	000b	RsvdP	Reserved.
4	0b	RWS	Data Link Protocol Error Mask
5	0b	RO	Reserved.
11:6	0x0	RsvdP	Reserved.
12	0b	RWS	Poisoned TLP Mask
13	0b	RWS	Flow Control Protocol Error Mask
14	0b	RWS	Completion Timeout Mask
15	0b	RWS	Completer Abort Mask
16	0b	RWS	Unexpected Completion Mask
17	0b	RWS	Receiver Overflow Mask
18	0b	RWS	Malformed TLP Mask
19	0b	RWS	ECRC Error Mask
20	0b	RWS	Unsupported Request Error Mask
21	0b	RO	ACS Violation Mask
25:22	0x0	RO	Not Supported.
31:26	0x0	RsvdP	Reserved.

9.2.4.1.4 Uncorrectable Error Severity Register (0x10C; RWS)

The Uncorrectable Error Severity register controls whether an individual uncorrectable error is reported as a fatal error. An uncorrectable error is reported as fatal when the corresponding error bit in the severity register is set. If the bit is cleared, the corresponding error is considered non-fatal.

Bit(s)	Init.	Type	Description
0	0b	RO	Reserved.
3:1	000b	RsvdP	Reserved.
4	1b	RWS	Data Link Protocol Error Severity
5	0b	RO	Reserved.
11:6	0x0	RsvdP	Reserved.
12	0b	RWS	Poisoned TLP Severity
13	1b	RWS	Flow Control Protocol Error Severity
14	0b	RWS	Completion Timeout Severity
15	0b	RWS	Completer Abort Severity
16	0b	RWS	Unexpected Completion Severity
17	1b	RWS	Receiver Overflow Severity
18	1b	RWS	Malformed TLP Severity
19	0b	RWS	ECRC Error Severity
20	0b	RWS	Unsupported Request Error Severity

Bit(s)	Init.	Type	Description
21	0b	RO	ACS Violation Severity
25:22	0x0	RO	Not Supported.
31:26	0x0	RsvdP	Reserved.

9.2.4.1.5 Correctable Error Status Register (0x110; RW1CS)

The Correctable Error Status register reports error status of individual correctable error sources on a PCIe device. When an individual error status bit is set to 1b it indicates that a particular error occurred; software can clear an error status by writing a 1b to the respective bit. Register is cleared by LAN_PWR_GOOD.

Bit(s)	Init.	Type	Description
0	0b	RW1CS	Receiver Error Status
5:1	0x0	RsvdZ	Reserved.
6	0b	RW1CS	Bad TLP Status
7	0b	RW1CS	Bad DLLP Status
8	0b	RW1CS	REPLAY_NUM Rollover Status
11:9	000b	RsvdZ	Reserved.
12	0b	RW1CS	Replay Timer Timeout Status
13	0b	RW1CS	Advisory Non-Fatal Error Status
14	0b	RW1CS	Corrected Internal Error Status
15	0b	RW1CS	Header Log Overflow Status
31:16	00b	RO	Reserved.

9.2.4.1.6 Correctable Error Mask Register (0x114; RWS)

The Correctable Error Mask register controls reporting of individual correctable errors by device to the host bridge via a PCIe error message. A masked error (respective bit set in mask register) is not reported to the host bridge by an individual device. There is a mask bit per bit in the Correctable Error Status register.

Bit(s)	Init.	Type	Description
0	0b	RWS	Receiver Error Mask
5:1	0x0	RsvdP	Reserved.
6	0b	RWS	Bad TLP Mask
7	0b	RWS	Bad DLLP Mask
8	0b	RWS	REPLAY_NUM Rollover Mask
11:9	000b	RsvdP	Reserved.
12	0b	RWS	Replay Timer Timeout Mask
13	1b	RWS	Advisory Non-Fatal Error Mask This bit is set by default to enable compatibility with software that does not comprehend Role-Based Error Reporting.
14	1b	RWS	Corrected Internal Error Mask
15	1b	RWS	Header Log Overflow Mask
31:16	00b	RO	Reserved.

9.2.4.1.7 Advanced Error Capabilities and Control Register (0x118; RO)

Bit(s)	Init.	Type	Description
4:0	0x0	ROS	Vector Vector pointing to the first recorded error in the Uncorrectable Error Status register. This is a read-only field that identifies the bit position of the first uncorrectable error reported in the Uncorrectable Error Status register.
5	1b	RO	ECRC Generation Capable If set, this bit indicates that the function is capable of generating ECRC. This bit is loaded from NVM. It is reflected in the PCI_CAPSUP register.
6	0b	RWS	ECRC Generation Enable When set, ECRC generation is enabled.
7	1b	RO	ECRC Check Capable If set, this bit indicates that the function is capable of checking ECRC. This bit is loaded from NVM. It is reflected in the PCI_CAPSUP register.
8	0b	RWS	ECRC Check Enable When set Set, ECRC checking is enabled.
9	0b	RO	Multiple Header Recording Capable Not Supported. Hard-wired to 0b
10	0b	RO	Multiple Header Recording Enable Not Supported. Hard-wired to 0b
11	0b	RsvdP	TLP Prefix Log Present Not supported. Hard-wired to 0b
15:12	0x0	RsvdP	Reserved.

9.2.4.1.8 Header Log Register (0x11C:128; RO)

The header log register captures the header for the transaction that generated an error. This register is 16 bytes.

Bit(s)	Init.	Type	Description
127:0	0x0	ROS	Header Log Register Header of the packet in error (TLP or DLLP).

9.2.4.2 Serial Number Capability

The PCIe device serial number capability is an optional extended capability that can be implemented by any PCIe device. The device serial number is a read-only 64-bit value that is unique for a given PCIe device.

The E610 implements this capability on all the functions and returns the same value in both.

Table 9-26. Serial Number Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x148	Next Capability Offset		Version (0x1)	Serial ID Capability ID (0x0003)
0x14C	Serial Number Register (Lower DWord)			
0x150	Serial Number Register (Upper DWord)			

Table 9-27 summarizes configuration sharing of the Serial ID Capability registers among the different PCI functions.

Table 9-27. Configuration Sharing of the Serial Number Capability

Field	Sub-field	Shared?	Replicated?	Comments
Enhanced Capability Header Register	Extended Capability ID	x		
	Capability Version	x		
	Next Capability Offset		x	
Serial Number Registers		x		

9.2.4.2.1 Device Serial Number Enhanced Capability Header Register (0x148; RO)

Bit(s)	Init.	Type	Description
15:0	0x3	RO	PCIe Extended Capability ID This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability. The extended capability ID for the device serial number capability is 0x0003.
19:16	0x1	RO	Capability Version This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Note: Must be set to 0x1 for this version of the specification.
31:20	See description	RO	Next Capability Offset This field contains the offset to the next PCIe capability structure or 0x000 if no other items exist in the linked list of capabilities. See Table 9-23 and Table 9-12 for possible values of the next capability offset.

9.2.4.2.2 Serial Number Registers (0x14C:0x150; RO)

The Serial Number register is a 64-bit field that contains the IEEE defined 64-bit Extended Unique Identifier (EUI-64*). The register at offset 0x144 holds the lower 32 bits and the register at offset 0x148 holds the higher 32 bits. The following figure details the allocation of register fields in the Serial Number register.

Bit(s)	Type	Description
63:0	RO	PCIe Device Serial Number This field contains the IEEE defined 64-bit EUI-64*. This identifier includes a 24-bit company ID value assigned by IEEE registration authority and a 40-bit extension identifier assigned by the manufacturer.

The serial number uses the Ethernet MAC Address according to the following definition:

Field	Company ID			Extension Identifier				
Order	Addr+0	Addr+1	Addr+2	Addr+3	Addr+4	Addr+5	Addr+6	Addr+7
Most Significant Byte						Least Significant Byte		
Most Significant Bit						Least Significant Bit		

The serial number can be constructed from the 48-bit Ethernet MAC Address in the following form:

Field	Company ID			MAC Label		Extension identifier		
Order	Addr+0	Addr+1	Addr+2	Addr+3	Addr+4	Addr+5	Addr+6	Addr+7
Most Significant Bytes						Least Significant Byte		
Most Significant Bit						Least Significant Bit		

In this case, the MAC label is 0xFFFF.

For example, assume that the company ID is (Intel) 00-A0-C9 and the extension identifier is 23-45-67. In this case, the 64-bit serial number is:

Field	Company ID			MAC Label		Extension Identifier		
Order	Addr+0	Addr+1	Addr+2	Addr+3	Addr+4	Addr+5	Addr+6	Addr+7
	00	A0	C9	FF	FF	23	45	67
Most Significant Byte						Least Significant Byte		
Most Significant Bit						Least Significant Bit		

The Ethernet MAC Address for the serial number capability is loaded from the NVM (not the same field that is loaded from NVM into the *RAL* and *RAH* registers). It is reflected in the *PCI_SERL* and *PCI_SERH* registers. The default value in case of no NVM is 0x0. In the above example:

- *PCI_SERL* = C9-23-45-67
- *PCI_SERH* = 00-00-00-A0

Note: The official document that defines EUI-64* is:
<http://standards.ieee.org/regauth/oui/tutorials/EUI64.html>

9.2.4.3 Alternate Routing ID Interpretation (ARI) Capability

To allow more than eight functions per endpoint without requesting an internal switch, as is usually needed in virtualization scenarios, the PCI-SIG defines a new capability that allows a different interpretation of the *Bus*, *Device*, and *Function* fields. The capability is exposed when the *PCI_CAPSUP.ARI_EN* bit is set from NVM.

The ARI capability structure is as follows:

Table 9-28. ARI Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x158	Next Capability Offset		ARI Capability ID (0x000E)	
0x15C	ARI Control Register		ARI Capabilities	

Table 9-29 summarizes configuration sharing of the ARI Capability registers among the different PCI functions.

Table 9-29. Configuration Sharing of the ARI Capability

Field	Sub-field	Shared?	Replicated?	Comments
Enhanced Capability Header Register	Extended Capability ID	x		
	Capability Version	x		
	Next Capability Offset		x	
ARI capability Register	Next Function Pointer		x	

9.2.4.3.1 PCIe ARI Header Register (0x158; RO)

Field	Bit(s)	Init.	Type	Description
ID	15:0	0xE	RO	PCIe Extended Capability ID PCIe extended capability ID for the alternative RID interpretation.
Version	19:16	0x1	RO	Capability Version This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 0x1 for this version of the specification.
Next Capability Offset	31:20	See description	RO	Next Capability Offset This field contains the offset to the next PCIe extended capability structure. See Table 9-23 . and Table 9-12 for possible values of the next capability offset.

9.2.4.3.2 PCIe ARI Capabilities and Control Register (0x15C; RO)

Field	Bit(s)	Init.	Type	Description
Reserved	7:0	0x2	RO	Reserved.
NFP	15:8	0x1 (func 0) 0x0 (func 1) ¹	RO	Next Function Pointer This field contains the pointer to the next physical function configuration space or 0x0000 if no other items exist in the linked list of functions. Function 0 is the start of the link list of functions.
Reserved	31:16	0x0	RO	Reserved.

1. Even if port 0 and port 1 are switched or function zero is a dummy function, this register should keep its attributes according to the function number. If LAN1 is disabled, the value of this field in function zero should be zero.

9.2.4.4 SR-IOV Capability

Note: This capability structure is not exposed in a dummy function.

This is a structure used to support the SR-IOV capabilities reporting and control. The capability is exposed when the PCI_CAPSUP.IOV_EN bit is set from NVM and the PCI_CNF2.NUM_VFS is non zero.

Table 9-30. IOV Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x1C4	Next Capability Offset (0x0)		Version (0x1)	IOV Capability ID (0x0010)
0x1C8	SR IOV Capabilities			
0x1CC	SR IOV Status			SR IOV Control
0x1D0	Total VFs (RO)			Initial VF (RO)

Table 9-30. IOV Capability Structure [continued]

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x1D4	Reserved	Function Dependency Link (RO)	Num VF (RW)	
0x1D8	VF Stride (RO)		First VF Offset (RO)	
0x1DC	VF Device ID		Reserved	
0x1E0	Supported Page Size (0x553)			
0x1E4	System Page Size (RW)			
0x1E8	VF BAR0 — Low (RW)			
0x1EC	VF BAR0 — High (RW)			
0x1F0	VF BAR2 (RO)			
0x1F4	VF BAR3 (RO)			
0x1F8	VF BAR4 — Low (RW)			
0x1FC	VF BAR4 — High (RW)			
0x200	VF Migration State Array Offset (RO)			

Table 9-31 summarizes configuration sharing of the SR-IOV Capability registers among the different PCI functions.

Table 9-31. Configuration Sharing of the SR-IOV Capability

Field	Sub-field	Shared?	Replicated?	Comments
Enhanced Capability Header Register	Extended Capability ID	x		
	Capability Version	x		
	Next Capability Offset		x	
SR-IOV Capabilities	VF Migration Capable	x		Not supported.
	ARI Capable Hierarchy Preserved		x	PF0 only. RO zero in all other functions.
	VF Migration Interrupt Message Number			Not supported.
SR-IOV Control	VF Enable		x	
	Memory Space Enable		x	
	ARI Capable Hierarchy	x		PF0 only. RO zero in all other functions.
InitialVFs			x	
TotalVFs			x	
NumVFs			x	
Function Dependency Link			x	Each PF indicates its PF number here.
First VF Offset			x	
VF Stride			x	
VF Device ID			x	
Supported Page Size		x		
System Page Size			x	
VF BARs			x	

9.2.4.4.1 PCIe SR-IOV Header Register (0x1C4; RO)

Field	Bit(s)	Init.	Type	Description
ID	15:0	0x10	RO	PCIe Extended Capability ID PCIe extended capability ID for the SR-IOV capability.
Version	19:16	0x1	RO	Capability Version This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 0x1 for this version of the specification.
Next offset	31:20	See description	RO	Next Capability Offset This field contains the offset to the next PCIe extended capability structure or 0x000 if no other items exist in the linked list of capabilities. See Table 9-23 . and Table 9-12 for possible values of the next capability offset.

9.2.4.4.2 PCIe SR-IOV Capabilities Register (0x1C8; RO)

Field	Bit(s)	Init.	Type	Description
Reserved	0	0b	RO	Reserved.
ARICHCP	1	1b/0b ¹	RO	ARI Capable Hierarchy Preserved If set, the ARI Capable Hierarchy bit is preserved across certain power state transitions.
Reserved	31:2	0x0	RO	Reserved.

1. Set on first function where SR-IOV is enabled (see `PCI_CAPSUP.IOV_EN` bit) and Read Only Zero in the other PF.

9.2.4.4.3 PCIe SR-IOV Control/Status Register (0x1CC; RW)

Field	Bit(s)	Init.	Type	Description
VFE	0	0b	RW	VF Enable/Disable VF Enable manages the assignment of VFs to the associated PF. If <i>VF Enable</i> is set to 1b, VFs must be enabled, associated with the PF, and exists in the PCIe fabric. When enabled, VFs must respond to and can issue PCIe transactions following all other rules for PCIe functions. If set to 0b, VFs must be disabled and not visible in the PCIe fabric; VFs cannot respond to or issue PCIe transactions. In addition, if <i>VF Enable</i> is cleared after having been set, all of the VFs must no longer: - Issue PCIe transactions - Respond to configuration space or memory space accesses. The behavior must be as if an FLR was issued to each of the VFs. Specifically, VFs must not retain any context after <i>VF Enable</i> has been cleared. Any errors already logged via PF error reporting registers, remain logged. However, no new VF errors must be logged after <i>VF Enable</i> is cleared.
Reserved	2:1	00b	RO	Reserved.
VF MSE	3	0b	RW	Memory Space Enable for Virtual Functions VF MSE controls memory space enable for all VFs associated with this PF as with the <i>Memory Space Enable</i> bit in a functions PCI command register. The default value for this bit is 0b. When VF Enable is 1, virtual function memory space access is permitted only when VF MSE is Set. VFs must follow the same error reporting rules as defined in the base specification if an attempt is made to access a virtual functions memory space when VF Enable is 1 and VF MSE is zero. Implementation Note: Virtual functions memory space cannot be accessed when VF Enable is zero. Thus, VF MSE is "don't care" when VF Enable is zero, however, software may choose to set VF MSE after programming the VF BARn registers, prior to setting VF Enable to 1.

Field	Bit(s)	Init.	Type	Description
ARI Capable Hierarchy	4	0b	RW (first function where SR-IOV is enabled) ROS (other function) ¹	ARI Capable Hierarchy The E610 is permitted to locate VFs in function numbers 8 to 255 of the captured bus number. This field is R/W in the lowest numbered PF. Other functions use the PF0 value as sticky. Note: If either ARI Capable Hierarchy Preserved is set (see Section 9.2.4.4.2) or <i>No_Soft_Reset</i> is set, a power state transition of this PF from D3hot to D0 does not affect the value of this bit.
Reserved	31:5	0x0	RO	Reserved.

1. Even if port 0 and port 1 are switched or function zero is a dummy function, this field should keep its attributes according to the function number.

9.2.4.4.4 PCIe SR-IOV Max/Total VFs Register (0x1D0; RO)

Field	Bit(s)	Init.	Type	Description
InitialVFs	15:0	64	RO	InitialVFs Indicates the number of VFs that are initially associated with the PF. If <i>VF Migration Capable</i> is cleared, this field must contain the same value as TotalVFs. In the E610 this parameter is equal to the TotalVFs in this register.
TotalVFs	31:16	64	RO	TotalVFs Defines the maximum number of VFs that can be associated with the PF. This field is loaded from the Max VFs field in the NVM. Reflected in <code>PCI_CNFG2.TOTAL_VFS</code> .

9.2.4.4.5 PCIe SR-IOV Num VFs Register (0x1D4; RW)

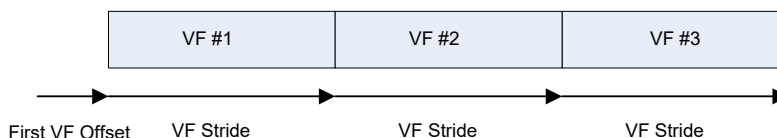
Field	Bit(s)	Init.	Type	Description
NumVFs	15:0	0x0	RW	Num VFs Defines the number of VFs software has assigned to the PF. Software sets NumVFs to any value between one and the TotalVFs as part of the process of creating VFs. NumVFs VFs must be visible in the PCIe fabric after both NumVFs is set to a valid value <i>VF Enable</i> is set to 1b.
FDL	23:16	0x0 (func 0) 0x1 (func 1) ¹	RO	Function Dependency Link Defines dependencies between physical functions allocation. In the E610 there are no constraints.
Reserved	31:24	0x0	RO	Reserved.

1. Even if port 0 and port 1 are switched or function zero is a dummy function, this register should keep its attributes according to the function number.

9.2.4.4.6 PCIe SR-IOV VF RID Mapping Register (0x1D8; RO)

Field	Bit(s)	Init.	Type	Description
FVO	15:0	0x180	RO	First VF offset Defines the requestor ID (RID) offset of the first VF that is associated with the PF that contains this capability structure. The first VFs 16-bit RID is calculated by adding the contents of this field to the RID of the PF containing this field. The content of this field is valid only when <i>VF Enable</i> is set. If <i>VF Enable</i> is 0b, the contents are undefined. If the <i>ARI Enable</i> bit is set, this field changes to 0x80.
VFS	31:16	0x2 ¹	RO	VF stride Defines the requestor ID (RID) offset from one VF to the next one for all VFs associated with the PF that contains this capability structure. The next VFs 16-bit RID is calculated by adding the contents of this field to the RID of the current VF. The contents of this field is valid only when <i>VF Enable</i> is set and <i>NumVFs</i> is non-zero. If <i>VF Enable</i> is 0b or if <i>NumVFs</i> is zero, the contents are undefined.

1. See [Section 7.8.2.6.1](#).



9.2.4.4.7 PCIe SR-IOV VF Device ID Register (0x1DC; RO)

Field	Bit(s)	Init.	Type	Description
DEVID	31:16	0x57AD	RO	VF Device ID This field contains the device ID that should be presented for every VF to the Virtual Machine (VM). The value of this field can be read from the <i>PCI_VFDEVID</i> in the NVM.
Reserved	15:0	0x0	RO	Reserved.

9.2.4.4.8 PCIe SR-IOV Supported Page Size Register (0x1E0; RO)

Field	Bit(s)	Init.	Type	Description
Supported Page Size	31:0	0x553	RO	Supported Page Size For PFs that supports the stride-based BAR mechanism, this field defines the supported page sizes. This PF supports a page size of $2^{(n+12)}$ if bit <i>n</i> is set. For example, if bit 0 is Set, the Endpoint (EP) supports 4KB page sizes. Endpoints are required to support 4 KB, 8 KB, 64 KB, 256 KB, 1 MB and 4 MB page sizes. All other page sizes are optional.

9.2.4.4.9 PCIe SR-IOV System Page Size Register (0x1E4; RW)

Field	Bit(s)	Init.	Type	Description
Page Size	31:0	0x1	RW	Page Size This field defines the page size the system uses to map the PF's and associated VFs' memory addresses. Software must set the value of the <i>System Page Size</i> to one of the page sizes set in the <i>Supported Page Sizes</i> field. As with <i>Supported Page Sizes</i>, if bit <i>n</i> is set in <i>System Page Size</i>, the PF and its associated VFs are required to support a page size of $2^{(n+12)}$. For example, if bit 1 is set, the system is using an 8 KB page size. The results are undefined if more than one bit is set in <i>System Page Size</i>. The results are undefined if a bit is set in <i>System Page Size</i> that is not set in <i>Supported Page Sizes</i>. When <i>System Page Size</i> is set, the PF and associated VFs are required to align all BAR resources on a <i>System Page Size</i> boundary. Each BAR size, including <i>VF BAR_n Size</i> (described later) must be aligned on a <i>System Page Size</i> boundary. Each BAR size, including <i>VF BAR_n Size</i> must be sized to consume a multiple of <i>System Page Size</i> bytes. All fields requiring page size alignment within a function must be aligned on a <i>System Page Size</i> boundary. <i>VF Enable</i> must be zero when <i>System Page Size</i> is set. The results are undefined if <i>System Page Size</i> is set when <i>VF Enable</i> is set.

9.2.4.4.10 PCIe SR-IOV BAR 0 — Low Register (0x1E8; RW)

Field	Bit(s)	Init.	Type	Description
Mem	0	0b	RO	Memory 0b = Indicates memory space.
Mem Type	2:1	10b	RO	Memory Type Indicates the address space size. 10b = 64-bit.
Prefetch Mem	3	0b*	RO	Prefetch Memory 0b = Non-prefetchable space. 1b = Prefetchable space. This bit is loaded from the NVM. It is reflected in the <i>PCI_VFSUP</i> register.
Memory Address Space	31:4	0x0	RW	Memory Address Space Which bits are RW bits and which are RO to 0x0 depend on the memory mapping window size. The size is a maximum between 16 KB and the page size.

9.2.4.4.11 PCIe SR-IOV BAR 0 — High Register (0x1EC; RW)

Field	Bit(s)	Init.	Type	Description
BAR0 — MSB	31:0	0x0	RW	MSB part of BAR0.

9.2.4.4.12 PCIe SR-IOV BAR 2 Register (0x1F0; RO)

Field	Bit(s)	Init.	Type	Description
BAR2	31:0	0x0	RO	This BAR is not used.

9.2.4.4.13 PCIe SR-IOV BAR 3 Register (0x1F4; RO)

Field	Bit(s)	Init.	Type	Description
BAR2	31:0	0x0	RO	This BAR is not used.

9.2.4.4.14 PCIe SR-IOV BAR 4 — Low Register (0x1F8; RW)

Field	Bit(s)	Init.	Type	Description
Mem	0	0b	RO	Memory 0b = Indicates memory space.
Mem Type	2:1	10b	RO	Memory Type Indicates the address space size. 10b = 64-bit.
Prefetch Mem	3	0b*	RO	Prefetch Memory 0b = Non-prefetchable space. 1b = Prefetchable space. This bit is loaded from the NVM. It is reflected in the PCI_VFSUP register.
Memory Address Space	31:4	0x0	RW	Memory Address Space Which bits are RW bits and which are RO to 0x0 depend on the memory mapping window size. The size is a maximum between 16 KB and the page size.

9.2.4.4.15 PCIe SR-IOV BAR 4 — High Register (0x1FC; RW)

Field	Bit(s)	Init.	Type	Description
BAR4 — MSB	31:0	0x0	RW	MSB part of BAR4.

9.2.4.4.16 PCIe SR-IOV VF Migration State Array Offset Register (0x200; RO)

Field	Bit(s)	Init.	Type	Description
Reserved	31:0	0x0	RO	Reserved.

9.2.4.5 TPH Requester Capability

The TPH Requester capability is an optional extended capability to support TLP Processing Hints. Table 9-32 lists the TPH extended capability structure for PCIe devices.

Table 9-32. TPH Requester Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0	
0x204	Next Capability Offset		Version (0x1)	TPH Capability ID (0x0017)	
0x208	TPH Requester Capability Register				
0x20C	TPH Requester Control Register				

Table 9.2.4.5.3 summarizes configuration sharing of the TPH Requester Capability registers among the different PCI functions.

Table 9-33. Configuration Sharing of the TPH Requester Capability

Field	Sub-field	Shared?	Replicated?	Comments
Enhanced Capability Header Register	Extended Capability ID	x		
	Capability Version	x		
	Next Capability Offset		x	
TPH Requester Capability		x		
TPH Requester Control			x	
TPH ST Table			x	The Steering Table Upper fields are not supported

9.2.4.5.1 TPH Requester Extended Capability Header (0x204; RO)

Bit(s)	Init.	Type	Description
15:0	0x17	RO	Extended Capability ID PCIe extended capability ID indicating TPH capability.
19:16	0x1	RO	Capability Version PCIe TPH extended capability version number.
31:20	See description	RO	Next Capability Offset This field contains the offset to the next PCIe capability structure. See Table 9-23. and Table 9-12 for possible values of the next capability offset.

9.2.4.5.2 TPH Requester Capability Register (0x208; RO)

Bit(s)	Init.	Type	Description
0	1b	RO	No ST Mode Supported If set indicates that the Function supports the No ST Mode of operation
1	0b	RO	Interrupt Vector Mode Supported Cleared to indicates that the E610 does not support Interrupt Vector Mode of operation
2	1b	RO	Device Specific Mode Set to indicate that the E610 supports Device Specific Mode of operation
7:3	0x0	RO	Reserved.
8	0b	RO	Extended TPH Requester Supported Cleared to indicate that the function is not capable of generating requests with Extended TPH TLP Prefix
10:9	00b	RO	ST Table Location Value indicates if and where the ST Table is located. Defined Encodings are: 00b = ST Table is not present. 01b = ST Table is located in the TPH Requester Capability structure. 10b = ST Table is located in the MSI-X Table structure. 11b = Reserved ST Table is not supported.
15:11	0x0	RO	Reserved.
26:16	0x0	RO	ST_Table Size System software reads this field to determine the ST_Table_Size N, which is encoded as N-1. For example, a returned value of 00000000011b indicates a table size of 4. The value in this field is undefined since the E610 does not support an ST Table
31:27	0x0	RO	Reserved.

9.2.4.5.3 TPH Requester Control Register (0x20C; R/W)

Bit(s)	Init.	Type	Description
2:0	000b	RW	ST Mode Select Indicates the ST mode of operation selected. Defined encodings are: 000b = No Table Mode 001b = Interrupt Vector Mode (not supported by the E610) 010b = Device Specific Mode All other values are reserved. The default value of 000 indicates No Table mode of operation. Functions that support only the No ST Mode of operation must hard-wire this field to 000b.
7:3	0x0	RO	Reserved.
9:8	00b	RW	TPH Requester Enable Controls the ability to issue Request TLPs using either TPH or Extended TPH. Defined Encodings are: 00b = The E610 is not permitted to issue transactions with TPH or Extended TPH as Requester 01b = The E610 is permitted to issue transactions with TPH as Requester and is not permitted to issue transactions with Extended TPH as Requester 10b = Reserved 11b = The E610 is permitted to issue transactions with TPH and Extended TPH as Requester (The E610 does not issue transactions with Extended TPH). The default value of this field is 00b.
31:10	0x0	RO	Reserved.

9.2.4.6 Access Control Services (ACS) Capability

The PCIe ACS defines a set of control points within a PCIe topology to determine whether a TLP should be routed normally, blocked, or redirected. ACS is applicable to RCs, switches, and multifunction devices and is not exposed in Single function mode (when function 1 is disabled). The capability is exposed when the PCI_CAPSUP.ACS_EN bit is set from NVM.

The ACS Capability structure is shared and exposed to all PFs.

Table 9-34 lists the PCIe ACS extended capability structure for PCIe devices.

Table 9-34. ACS Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x290	Next Capability Offset	Version (0x1)	ACS Capability ID (0x0D)	
0x294	ACS Control Register (0x0)		ACS Capability Register (0x0)	

9.2.4.6.1 ACS CAP ID (0x290; RO)

Bit(s)	Init.	Type	Description
15:0	0x0D	RO	ACS Capability ID PCIe extended capability ID indicating ACS capability.
19:16	0x1	RO	Version Number PCIe ACS extended capability version number.
31:20	See description	RO	Next Capability Pointer This is the last capability, so the next pointer is 0x000. See Table 9-23. and Table 9-12 for possible values of the next capability offset.

9.2.4.6.2 ACS Control and Capabilities (0x294; RO)

Bit(s)	Init.	Type	Description
0	0b	RO	ACS Source Validation (V) Hard-wired to zero, not supported in the E610.
1	0b	RO	ACS Translation Blocking (B) Hard-wired to zero, not supported in the E610.
2	0b	RO	ACS P2P Request Redirect (R) Hard-wired to zero, not supported in the E610.
3	0b	RO	ACS P2P Completion Redirect (C) Hard-wired to zero, not supported in the E610.
4	0b	RO	ACS Upstream Forwarding (U) Hard-wired to zero, not supported in the E610.
5	0b	RO	ACS P2P Egress Control (E) Hard-wired to zero, not supported in the E610.
6	0b	RO	ACS Direct Translated P2P (T) Hard-wired to zero, not supported in the E610.
7	0b	Rsrv	Reserved.
15:8	0x0	RO	Egress Control Vector Size Hard-wired to zero, not supported in the E610.
16	0b	RO	ACS Source Validation Enable (V) Hard-wired to zero, not supported in the E610.
17	0b	RO	ACS Translation Blocking Enable (B) Hard-wired to zero, not supported in the E610.
18	0b	RO	ACS P2P Request Redirect Enable (R) Hard-wired to zero, not supported in the E610.
19	0b	RO	ACS P2P Completion Redirect Enable (C) Hard-wired to zero, not supported in the E610.
20	0b	RO	ACS Upstream Forwarding Enable (U) Hard-wired to zero, not supported in the E610.
21	0b	RO	ACS P2P Egress Control Enable (E) Hard-wired to zero, not supported in the E610.
22	0b	RO	ACS Direct Translated P2P Enable (T) Hard-wired to zero, not supported in the E610.
31:23	0x0	Rsrv	Reserved.

9.2.4.7 Latency Tolerance Reporting (LTR) Capability

The Latency Tolerance Reporting Capability is an optional Extended Capability that allows software to provide platform latency information to devices with upstream ports (Endpoints and Switches). This capability structure is required if the device supports Latency Tolerance Reporting (LTR). The capability is exposed when the PCI_CAPSUP.LTR_EN bit is set from NVM.

The LTR Capability structure is implemented only in Function 0 even when Function 0 is a dummy function, and controls the component's Link behavior on behalf of all the Functions of the device

Table 9-35 lists the PCIe LTR extended capability structure for PCIe devices.

Table 9-35. LTR Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x298	PCI Express Extended Capability Header			
0x29C	Max No-Snoop Latency Register		Max Snoop Latency Register	

9.2.4.7.1 LTR Extended Capability Header (0x298; RO)

Bit(s)	Init.	Type	Description
15:0	0x18	RO	PCI Express Extended Capability ID PCIe extended capability ID indicating LTR capability.
19:16	0x1	RO	Capability Version PCIe LTR extended capability version number.
31:20	See description	RO	Next Capability Offset See Table 9-23 and Table 9-12 for possible values of the next capability offset.

9.2.4.7.2 Max Snoop Latency Register (0x29C; RW)

Bit(s)	Init.	Type	Description
9:0	0x0	RW	Maximum Snoop Latency Value Along with the <i>Max Snoop Latency Scale field</i> , this register specifies the maximum snoop latency that a device is permitted to request. Software should set this to the platform's maximum supported latency or less. Field is also an indicator of the platform maximum latency, should an endpoint send up LTR Latency Values with the <i>Requirement</i> bit not set.
12:10	000b	RW	Max Snoop Latency Scale This field provides a scale for the value contained within the <i>Maximum Snoop Latency Value</i> field. Encoding: 000b = Value times 1 ns 001b = Value times 32 ns 010b = Value times 1,024 ns 011b = Value times 32,768 ns 100b = Value times 1,048,576 ns 101b = Value times 33,554,432 ns 110b = Not permitted 111b = Not permitted
15:13	0x0	RW	Reserved.

9.2.4.7.3 Max No-Snoop Latency Register (0x29C; RW)

Bit(s)	Init.	Type	Description
9:0	0x0	RW	Maximum No-Snoop Latency Value Along with the <i>Max No-Snoop Latency Scale</i> field, this register specifies the maximum no-snoop latency that a device is permitted to request. Software should set this to the platform's maximum supported latency or less. Field is also an indicator of the platform's maximum latency, should an endpoint send up LTR Latency Values with the <i>Requirement</i> bit not set.
12:10	000b	RW	Max No-Snoop Latency Scale This field provides a scale for the value contained within the <i>Maximum No-Snoop Latency Value</i> field. Encoding: 000b = Value times 1 ns 001b = Value times 32 ns 010b = Value times 1,024 ns 011b = Value times 32,768 ns 100b = Value times 1,048,576 ns 110b = Not permitted 111b = Not permitted
15:13	0x0	RW	Reserved.

9.2.4.8 Secondary PCI Express Extended Capability

The Secondary PCI Express Extended Capability structure is required for all Ports and RCRBs that support a Link speed of 8.0 GT/s or higher. For Multi-Function Upstream Ports, this capability must be implemented in Function 0 and must not be implemented in other Functions. The capability is exposed when the PCI_CAPSUP.SEC_EN bit is set from NVM.

Table 9-36 lists the Secondary PCI Express extended capability structure for PCIe devices.

Table 9-36. Secondary PCI Express Extended Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x168	PCI Express Extended Capability Header			
0x16C	Link Control 3 Register			
0x170	Lane Error Status Register			
0x174	Equalization Control Register (Sized by Maximum Link Width)			
0x178	Equalization Control Register (Sized by Maximum Link Width)			

9.2.4.8.1 Secondary PCIe Extended Capability Header (0x168)

Bit(s)	Init.	Type	Description
15:0	0x19	RO	PCI Express Extended Capability ID This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. PCI Express Extended Capability ID for the Secondary PCI Express Extended Capability is 0019h.
19:16	0x1	RO	Capability Version This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.

Bit(s)	Init.	Type	Description
31:20	See description	RO	Next Capability Offset See Table 9-23. and Table 9-12 for possible values of the next capability offset.

9.2.4.8.2 Link Control 3 Register (0x16C)

Bit(s)	Init.	Type	Description
0	0b	RW/RsvdP	Perform Equalization When this bit is 1b and a 1b is written to the Link Retrain register with Target Link Speed set to 8.0 GT/s, the Downstream Port must perform Transmitter Equalization. This bit is RW for Upstream Ports when <i>Crosslink Supported</i> is 1b. This bit is not applicable and is RsvdP for Upstream Ports when the <i>Crosslink Supported</i> bit is 0b. The default value is 0b.
1	0b	RW/RsvdP	Link Equalization Request Interrupt Enable When Set, this bit enables the generation of interrupt to indicate that the <i>Link Equalization Request</i> bit has been set. This bit is RW for Upstream Ports when <i>Crosslink Supported</i> is 1b. This bit is not applicable and is RsvdP for Upstream Ports when the <i>Crosslink Supported</i> bit is 0b. The default value for this bit is 0b.
31:2	0x0	RsvdZ	Reserved.

9.2.4.8.3 Lane Error Status Register (0x170)

The Lane Error Status register consists of a 32-bit vector, where each bit indicates if the corresponding PCI Express Lane detected an error.

Bit(s)	Init.	Type	Description
7:0	0x0	RW1CS	Lane Error Status Bits Each bit indicates if the corresponding Lane detected a Lane-based error. A value of 1b indicates that a Lane based-error was detected on the corresponding Lane Number. The default value of this field is 0b. This field is intended for debug purposes only.
31:8	0x0	RsvdZ	Reserved.

9.2.4.8.4 Lane Equalization Control Register (0x174: 0x17B)

The Equalization Control register consists of control fields required for per Lane equalization and number of entries in this register are sized by Maximum Link Width.

15

0

Lane (0) Equalization Control Register
Lane (1) Equalization Control Register
...
Lane (Maximum Link Width - 1) Equalization Control Register

Lane ((Maximum Link Width – 1):0) Equalization Control Register:

Bit(s)	Init.	Type	Description
3:0	0x0	RsvdP	Downstream Port Transmitter Preset For an Upstream Port if Crosslink Supported is 0b, this field is RsvdP.
6:4	000b	RsvdP	Downstream Port Receiver Preset Hint For an Upstream Port if Crosslink Supported is 0b, this field is RsvdP.
11:8	0xF	RO	Upstream Port Transmitter Preset Field contains the Transmit Preset value sent or received during Link Equalization. Since crosslink is not supported, the field is intended for debug and diagnostics. It contains the value captured from the associated Lane during Link Equalization. Field is RO. The default value is 1111b.
14:12	111b	HwInit/RO	Upstream Port Receiver Preset Hint Field contains the Receiver Preset Hint value sent or received during Link Equalization. Field usage varies as follows: Since crosslink is not supported, the field is intended for debug and diagnostics. It contains the value captured from the associated Lane during Link Equalization. Field is RO. The default value is 111b.
15	0b	Rsvd	Reserved.

9.2.4.9 Data Link Feature Extended Capability

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x3D8	PCI Express Extended Capability Header			
0x3DC	Data Link Feature - Capabilities Register			
0x3E0	Data Link Feature - Status Register			

9.2.4.9.1 Data Link Feature Extended Capability Header (0x3D8; RO)

Bit(s)	Init.	Type	Description
15:0	0x25	RO	PCI Express Extended Capability ID This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. PCI Express Extended Capability ID for the Data Link Feature PCI Express Extended Capability is 0025h.
19:16	0x1	RO	Capability Version This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.
31:20	See description	RO	Next Capability Offset See Table 9-23 and Table 9-12 for possible values of the next capability offset.

9.2.4.9.2 Data Link Feature Capabilities Register (0x3DC; RO)

Bit(s)	Init.	Type	Description
22:0	0x0000 1	RO	Local Data Link Feature Supported This field contains the Feature Supported value used when this Port sends a Data Link Feature DLLP. Define features are: Bit 0 - Local Scaled Flow Control Supported This bit indicates that the Port supports the Scaled Flow Control Feature Bits 22:1 - RsvdP
30:23	0x0	RO	Reserved
31	0x1	RO	Data Link Feature Exchange Enable If set, this bit indicates that this Port will enter the DL_Feature negotiation state

9.2.4.9.3 Data Link Feature Status Register (0x3E0, RO)

Bit(s)	Init.	Type	Description
22:0	0x0000 0	RO	Remote Data Link Data Link Feature Supported These bits indicate that the Remote Port supports the corresponding Data Link Feature. These bits capture all information from the Feature Supported field of the Data Link Feature DLLP even when this Port doesn't support the corresponding feature. This field is Cleared on entry to state DL_Inactive. Features currently defined are: Bit 0 - Remote Scaled Flow Control Supported This bit indicates that the Remote Port supports the Scaled Flow Control Feature Bits 22:1 - Undefined
23	0x0	RO	Remote Data Link Feature Ack This bit indicates that the Remote Port has received this Port's Data Link Feature DLLP. This bit captures the Feature Ack bit of the Data Link Feature DLLP. This bit is Cleared on entry to state DL_Inactive
30:24	0x0	RO	Reserved
31	0x1	RO	Remote Data Link Feature Supported Valid This bit indicates that the Port has received a Data Link Feature DLLP in state DL_Feature and that the Remote Data Link Feature Supported and Remote Data Link Feature Ack fields are meaningful. This bit is Cleared on entry to state DL_Inactive

9.2.4.10 Physical Layer 16.0 GT/s Capability

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x188	Physical Layer 16.0 GT/s Extended Capability Header			
0x18C	Physical Layer 16.0 GT/s - Capabilities Register			
0x190	Physical Layer 16.0 GT/s - Control Register			
0x194	Physical Layer 16.0 GT/s - Status Register			
0x198	16.0 GT/s Local Data Parity Mismatch Status Register			
0x19C	16.0 GT/s First Retimer Data Parity Mismatch Status Register			
0x1A0	16.0 GT/s Second Retimer Data Parity Mismatch Status Register			
0x1A4	Spare			
16.0 GT/s Lane Equalization Control Registers				
0x1A8	Lane 3	Lane 2	Lane 1	Lane 0

9.2.4.10.1 Physical Layer 16.0 GT/s Extended Capability Header (0x188; RO)

Bit(s)	Init.	Type	Description
15:0	0x26	RO	PCI Express Extended Capability ID This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. PCI Express Extended Capability ID for the Physical Layer 16.0 GT/s PCI Express Extended Capability is 0026h.
19:16	0x1	RO	Capability Version This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.
31:20	See description	RO	Next Capability Offset See Table 9-23. and Table 9-12 for possible values of the next capability offset.

9.2.4.10.2 Physical Layer 16.0 GT/s Capabilities Register (0x18C; RsvdP)

Bit(s)	Init.	Type	Description
31:0	0x0	RsvdP	Reserved

9.2.4.10.3 Physical Layer 16.0 GT/s Control Register (0x190; RsvdP)

Bit(s)	Init.	Type	Description
31:0	0x0	RsvdP	Reserved

9.2.4.10.4 Physical Layer 16.0 GT/s Status Register (0x194; RW1CS)

This register is valid only in function 0, and returns zero in other functions

Bit(s)	Init.	Type	Description
0	0x0	ROS	Equalization 16.0 GT/s Complete
1	0x0	ROS	Equalization 16.0 GT/s Phase 1 Successful
2	0x0	ROS	Equalization 16.0 GT/s Phase 2 Successful
3	0x0	ROS	Equalization 16.0 GT/s Phase 3 Successful
4	0x0	RW1CS	Link Equalization Request 16.0 GT/s
31:5	0x0	RsvdZ	Reserved

9.2.4.10.5 16.0 GT/s Local Data Parity Mismatch Status Register (0x198, RW1CS)

Bit(s)	Init.	Type	Description
3:0	0x0	RW1CS	Local Data Parity Mismatch Status Each bit indicates if the corresponding Lane detected a Data Parity mismatch. A value of 1b indicates that a mismatch was detected on the corresponding Lane
31:4	0x0	RsvdZ	Reserved

9.2.4.10.6 16.0 GT/s First Retimer Data Parity Mismatch Status Register (0x19C; RW1CS)

Bit(s)	Init.	Type	Description
3:0	0x0	RW1CS	First Retimer Data Parity Mismatch Status Each bit indicates if the corresponding Lane detected a Data Parity mismatch. A value of 1b indicates that a mismatch was detected on the corresponding Lane
31:4	0x0	RsvdZ	Reserved

9.2.4.10.7 16.0 GT/s Second Retimer Data Parity Mismatch Status Register (0x1A0; RW1CS)

Bit(s)	Init.	Type	Description
3:0	0x0	RW1CS	Second Retimer Data Parity Mismatch Status Each bit indicates if the corresponding Lane detected a Data Parity mismatch. A value of 1b indicates that a mismatch was detected on the corresponding Lane
31:4	0x0	RsvdZ	Reserved

9.2.4.10.8 Physical Layer 16.0 GT/s Status 2 Register (0x1A4; RsvdZ)

Bit(s)	Init.	Type	Description
31:0	0x0	RsvdZ	Reserved

9.2.4.10.9 16.0 GT/s Lane Equalization Control Register (0x1A8; HWInit)

This register defines the control fields required for per lane 16.0 GT/s equalization. A byte is allocated to each lane.

Each byte is defined as follows:

Bit(s)	Init.	Type	Description
3:0	0xF	HwInit	Downstream Port 16.0 GT/s Transmitter Preset
7:4	0xF	HwInit	Upstream Port 16.0 GT/s Transmitter Preset

9.2.4.11 Lane Margining at the Receiver Capability

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x1AC	Lane Margining at the Receiver Extended Capability Header			
0x1B0	Capabilities Register		Status Register	
0x1B4-1C0	Lane n - Status Register		Lane n - Control Register	

9.2.4.11.1 Lane Margining at the Receiver Capability Header (0x1AC; RO)

Bit(s)	Init.	Type	Description
15:0	0x27	RO	PCI Express Extended Capability ID This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. PCI Express Extended Capability ID for the Lane Margining at the Receiver Capability PCI Express Extended Capability is 0027h.
19:16	0x1	RO	Capability Version This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.
31:20	See description	RO	Next Capability Offset See Table 9-23 , and Table 9-12 for possible values of the next capability offset.

9.2.4.11.2 Lane Margining at the Receiver Capabilities Register (0x1B0; RO)

Bit(s)	Init.	Type	Description
0	HWInit	RO	Margining uses Driver Software
15:1	0x0	RsvdP	Reserved

9.2.4.11.3 Lane Margining at the Receiver Status Register (0x1B0; RO)

Bit(s)	Init.	Type	Description
0	HWInit	RO	Margining Ready
1	HWInit	RO	Margining Software Ready
15:2	0x0	RsvdP	Reserved

9.2.4.11.4 Margining Lane #n Control Register (0x1B4 + 4*n; RW)

Bit(s)	Init.	Type	Description
2:0	0x0	RW	Receiver Number
5:3	0x7	RW	Margin Type
6	0x0	RW	Usage Model
7	0x0	RO	Reserved

Bit(s)	Init.	Type	Description
15:8	0x9C	RW	Margin Payload

9.2.4.11.5 Margining Lane #n Status Register (0x1B6 + 4*n; RW)

Bit(s)	Init.	Type	Description
2:0	0x0	RO	Receiver Number Status
5:3	0x0	RO	Margin Type Status
6	0x0	RO	Usage Model Status
7	0x0	RO	Reserved
15:8	0x0	RO	Margin Payload Status

9.2.4.12 Data Object Exchange (DOE) Capability

The Data Object Exchange (DOE) Extended Capability is an optional Extended Capability for discovering and controlling a mechanism for the exchange of data objects.

DOE defines an optional Extended Capability structure and associated control mechanisms to provide System Firmware/Software the ability to perform data object exchanges with a PCIe function.

To support Component Measurement and Authentication (CMA) accessible in-band via host firmware/software, Data Object Exchange (DOE) is required.

This capability is exposed if the GLPCI_CAPSUP.DOE_EN bit is set.

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x580	DOE Extended Capability Header			
0x584	DOE - Capabilities Register			
0x588	DOE - Control Register			
0x58C	DOE - Status Register			
0x590	DOE Write Data Mailbox Register			
0x594	DOE Read Data Mailbox Register			

9.2.4.12.1 DOE Capability Header (0x580; RO)

Bit(s)	Init.	Type	Description
15:0	0x2E	RO	PCI Express Extended Capability ID This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. PCI Express Extended Capability ID for the DOE PCI Express Extended Capability is 002Eh.
19:16	0x1	RO	Capability Version This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.
31:20	See description	RO	Next Capability Offset See Table 9-23 . and Table 9-12 for possible values of the next capability offset.

9.2.4.12.2 DOE Capabilities Register (0x584; RO)

Bit(s)	Init.	Type	Description
0	0x0	RO	Interrupt Support When set, this bit indicates DOE support for using MSI/MSI-X to indicate the availability of a data object.
11:1	0x0	RO	DOE Interrupt Message Number When the Interrupt Support bit is Set, this field indicates which MSI/MSI-X vector is used for the interrupt message generated in association with DOE
31:12	0x0	RO	Reserved

9.2.4.12.3 DOE Control Register (0x588; RW)

Bit(s)	Init.	Type	Description
0	0x0	RW	DOE Abort A write of 1b to this bit must cause all data object transfer operation associated with this DOE instance to be aborted. Reads from this bit always return 0b.
1	0x0	RW	DOE Interrupt Enable When Set, and MSI/MSI-X is enabled, the DOE instance must issue an MSI/MSI-X interrupt
30:2	0x0	RO	Reserved
31	0x0	RW	DOE Go A write of 1b to this bit indicates to the DOE instance that it can start consuming the data object transferred through the DOE Write Data Mailbox register. Behavior is undefined if the DOE Go bit is set before the entire data object has been written to the DOE Write Data Mailbox register. Behavior is undefined if the DOE Go bit is written with 1b when the DOE Busy bit is Set. Reads from this bit always returns 0b.

9.2.4.12.4 DOE Status Register (0x58C; RW)

Bit(s)	Init.	Type	Description
0	0x1	RO	DOE Busy When Set, this bit indicates the DOE instance is temporarily unable to receive a new data object through the DOE Write Data Mailbox register. The DOE instance sets this bit when processing a received data object, and clears it when it is able to receive a new data object.
1	0x0	RW1C	DOE Interrupt Status This bit is set when an interrupt is generated to indicate that the Data Object Ready bit or the DOE Error bit has been Set, or that the DOE Busy bit has been Cleared.
2	0x0	RO	DOE Error When Set, this bit indicates that there has been an internal error associated with data object received, or that a data object has been received for which the DOE instance is unable to provide a response. The DOE instance must Clear this bit, if it is not already Clear, when 1b is written to the DOE Abort bit in the DOE Control Register. Writing 1b to the DOE Abort bit is the only mechanism for software to Clear this bit.
30:3	0x0	RO	Reserved
31	0x0	RO	Data Object Ready When Set, this bit indicates the DOE instance has a data object available to be read by system firmware/software. If there is no additional data object ready for transfer, the DOE instance clears this bit after the entire data object has been transferred, as indicated by software writing to the DOE Read Data Mailbox register after reading the final DW of the data object. The DOE instance clears this bit, upon a write of 1b to the DOE Abort bit in the DOE Control register.

9.2.4.12.5 DOE Write Data Mailbox Register (0x590; RW)

Bit(s)	Init.	Type	Description
31:0	0x0	RW	DOE Write Data Mailbox The DOE instance receives data objects via writes to this register. A successfully completed write to this register adds one DW to the incoming data object. Setting the DOE Go bit in the DOE Control Register indicates to the DOE Instance that the final DW of the data object has been written to this register.

9.2.4.12.6 DOE Read Data Mailbox Register (0x594; RW)

Bit(s)	Init.	Type	Description
31:0	0x0	RW	DOE Read Data Mailbox If the Data Object Ready bit is Set, a read of this register returns the current DW of the data object. A write of any value to this register indicates a successful transfer of the current data object DW, and the DOE instance returns the next DW in the data object upon the next read of this register as long as Data Object Ready bit remains Set. It is permitted for multiple data objects to be read from this register back-to-back. When this scenario occurs the Data Object Ready bit will remain Set until the final DW is read. A write of any value to this register when the Data Object Ready bit is Clear doesn't have any effect.

9.2.5 CSR Access Via Configuration Address Space

The registers described in the section are not part of the PCI Express standard configuration and can be used to access the CSR space before memory BARs are allocated. When this mechanism is used, there is no need to expose an I/O BAR for pre boot operation.

Note: These registers are not available for dummy functions.

9.2.5.1 IOADDR Register (0xF8; R/W)

This is a read/write register. Each function has its own IOADDR register. Functionality is the same in all functions. Register is cleared at power-up (LAN_PWR_GOOD) or PCIe reset.

Note: When functioning in a D3 state, software should not attempt to access CSRs via the IOADDR and IODATA registers.

Bit(s)	Init.	Type	Description
30:0	0x0	R/W ¹	Internal Register or Internal Memory location Address 0x00000-0x1FFFF = Internal registers and memories 0x20000-0x7FFFFFFF = Undefined
31	0b	R/W	Configuration IO Access Enable 0b = CSR configuration read or write disabled. 1b = CSR configuration read or write enabled. When this bit is set, accesses to the IODATA register actually generate transactions to the E610. Otherwise, accesses to the IODATA register are don't-cares (writes are discarded silently, reads return arbitrary results).

1. In the event that the PCI_CAPSUP.CSR_CONF_EN bit is cleared, accesses to the IOADDR register via the configuration address space is ignored and has no effect on the register and the CSRs referenced by the IOADDR register.

9.2.5.2 IODATA Register (0xFC; R/W)

This is a read/write register. Each function has its own IODATA register. Functionality is the same in all functions. Register is cleared at power-up (LAN_PWR_GOOD) or PCIe reset.

9.3 Virtual Functions Configuration Space

Bit(s)	Init.	Type	Description
31:0	0x0	R/W ¹	Data field for reads or writes to the Internal register or internal memory location as identified by the current value in IOADDR. All 32 bits of this register can be read from or written to.

1. In the event that the *IO_by_cfg* bit in the *PCIe Init Configuration 2* EEPROM word is cleared, access to the IODATA register via the configuration address space is ignored and has no effect on the register and the CSRs referenced by the IOADDR register.

The configuration space reflected to each of the VF is a sparse version of the physical function configuration space. [Table 9-37](#) describes the behavior of each register in the VF configuration space.

Table 9-37. VF PCIe Configuration Space

Section	Offset	Name	VF behavior	Notes
PCI Mandatory Registers	0	Vendor ID	RO — 0xFFFF	
	2	Device ID	RO — 0xFFFF	
	4	Command	Per VF	See Section 9.3.1.1 .
	6	Status	Per VF	See Section 9.3.1.2 .
	8	RevisionID	RO as PF	
	9	Class Code	RO as PF	
	C	Cache Line Size	RO — 0x0	
	D	Latency Timer	RO — 0x0	
	E	Header Type	RO — 0x0	
	F	Reserved	RO — 0x0	
	10 — 27	BARs	RO — 0x0	Emulated by VMM.
	28	CardBus CIS	RO — 0x0	Not used.
	2C	Sub Vendor ID	RO as PF	
	2E	Sub System	RO as PF	
	30	Expansion ROM	RO — 0x0	Emulated by VMM.
	34	Cap Pointer	RO — 0x70	Next = MSI-X capability.
	3C	Int Line	RO — 0x0	
	3D	Int Pin	RO — 0x0	
	3E	Max Lat/Min Gnt	RO — 0x0	
MSI-X Capability	B0	MSI-X Header	RO — 0x7011	Next = PCIe capability.
	B2	MSI-x Message Control	Per VF	See Section 9.3.2.1.1 .
	B4	MSI-X table Address	RO — as PF	See Section 9.3.2.1.2
	B8	MSI-X PBA Address	RO	See Section 9.3.2.1.3

Table 9-37. VF PCIe Configuration Space [continued]

Section	Offset	Name	VF behavior	Notes
PCIe Capability	70	PCIe Header	RO — 0x0010	Next = Last capability.
	72	PCIe Capabilities	RO — as PF	
	74	PCIe Dev Cap	RO — as PF	
	78	PCIe Dev Ctrl	RW	As PF apart from FLR — See Section 9.3.2.2.1.
	7A	PCIe Dev Status	Per VF	See Section 9.3.2.2.2.
	7C	PCIe Link Cap	RO — as PF	
	80	PCIe Link Ctrl	RO — 0x0	
	82	PCIe Link Status	RO — 0x0	
	84	PCIe Dev Cap 2	RO — as PF	
	88	PCIe Dev Ctrl 2	RO — 0x0	
	90	PCIe Link Ctrl 2	RO — 0x0	
	92	PCIe Link Status 2	RO — 0x0	
AER Capability	100	AER — Header	RO — 0x14810001	Next = ARI structure.
	104	AER — Uncorr Status	Per VF	See Section 9.3.3.1.1.
	108	AER — Uncorr Mask	RO — 0x0	
	10C	AER — Uncorr Severity	RO — 0x0	
	110	AER — Corr Status	Per VF	See Section 9.3.3.1.2.
	114	AER — Corr Mask	RO — 0x0	
	118	AER — Cap/Ctrl	RO as PF	
	11C — 128	AER — Error Log	Shared two logs for all VFs	Same structure as in PF. In case of overflow, the header log is filled with ones.
ARI Capability	148	ARI — Header	0x1581000E	Next = TPH.
	14C	ARI — Cap/Ctrl	RO — 0x0	
TPH Capability	158	TPH - Header	RO - 0x1E410017	Next = ACS.
	15C	TPH - Capability	RO - 0x00000005	No ST Table.
	160	TPH - Control	Per VF	Same structure as PF.
ACS capability	1E4	ACS - Header	RO - 0x0001000D	Last extended capability.
	1E8	ACS - Capability	RO - 0x00000000	

9.3.1 Mandatory Configuration Space

9.3.1.1 VF Command Register (0x4; RW)

Bit(s)	Init.	Type	Description
0	0b	RO	IOAE I/O Access Enable. RO as zero field.
1	0b	RO	MAE Memory Access Enable. RO as zero field.
2	0b	RW	BME Bus Master Enable. Disabling this bit prevents the associated VF from issuing any memory or I/O requests. Note that as MSI/MSI-X interrupt messages are in-band memory writes, disabling the bus master enable bit disables MSI/MSI-X interrupt messages as well. Requests other than memory or I/O requests are not controlled by this bit. Note: The state of active transactions is not specified when this bit is disabled after being enabled. The device can choose how it behaves when this condition occurs. Software cannot count on the device retaining state and resuming without loss of data when the bit is re-enabled. Transactions for a VF that has its <i>Bus Master Enable</i> set must not be blocked by transactions for VFs that have their <i>Bus Master Enable</i> cleared.
3	0b	RO	SCM Special Cycle Enable. Hard-wired to 0b
4	0b	RO	MWIE MWI Enable. Hard-wired to 0b.
5	0b	RO	PSE Palette Snoop Enable. Hard-wired to 0b.
6	0b	RO	PER Parity Error Response. Zero for VFs.
7	0b	RO	WCE: Wait Cycle Enable Hard-wired to 0b.
8	0b	RO	SERRE SERR# Enable. Zero for VFs.
9	0b	RO	FB2BE Fast Back-to-Back Enable. Hard-wired to 0b.
10	0b	RO	INTD Interrupt Disable. Hard-wired to 0b.
15:11	0x0	RO	Reserved.

9.3.1.2 VF Status Register (0x6; RW)

Bit(s)	Init.	Type	Description
2:0	0x0	RO	Reserved.
3	0b	RO	IS. Interrupt Status. Hard-wired to 0b.
4	1b	RO	NC New Capabilities. Indicates that the E610 VFs implement extended capabilities. E610 VFs implement a capabilities list, to indicate that it supports MSI-X and PCIe extensions.
5	0b	RO	66E 66 MHz Capable. Hard-wired to 0b.
6	0b	RO	Reserved.

Bit(s)	Init.	Type	Description
7	0b	RO	FB2BC Fast Back-to-Back Capable. Hard-wired to 0b.
8	0b	RW1C	MPERR Data Parity Reported.
10:9	00b	RO	DEVSEL DEVSEL Timing. Hard-wired to 0b.
11	0b	RW1C	STA Signaled Target Abort.
12	0b	RW1C	RTA Received Target Abort.
13	0b	RW1C	RMA Received Master Abort.
14	0b	RW1C	SSERR Signaled System Error.
15	0b	RW1C	DSERR Detected Parity Error.

9.3.2 PCI Capabilities

9.3.2.1 MSI-X Capability

The only registers with a different layout than the PF for MSI-X, is the control register.

Note: The message address and data registers in enhanced mode use the first MSI-X entry of each VF in the regular MSI-X table.

See [Section 8.2.2.7](#) for details of the MSI-X registers in BAR 4 of the VF.

9.3.2.1.1 VF MSI-X Control Register (0xB2; RW)

Bit(s)	Init.	Type	Description
10:0	0x2 ¹	RO	TS Table Size.
13:11	000b	RO	Reserved.
14	0b	RW	Mask Function Mask.
15	0b	RW	En MSI-X Enable.

1. Default value is read from the NVM. This field is loaded from the PCI_CNF2.MSI_X_VF_N register field.

9.3.2.1.2 MSI-X Table Offset Register (0xB4; RW)

Bit(s)	Init.	Type	Description
2:0	100b	RO	Table BIR Indicates which one of a function's BARs, beginning at 0x10 in the configuration space, is used to map the function's MSI-X table into the memory space. BIR values: 0...5 correspond to BARs 0x10...0x24, respectively. BIR equals 4 to indicate BAR 4 and 5 are used for MSI-X vectors.
31:3	0x0	RO	Table Offset Used as an offset from the address contained in one of the function's BARs to point to the base of the MSI-X table. The lower three <i>Table BIR</i> bits are masked off (set to 0b) by software to form a 32-bit QWord-aligned offset. This field is read only.

9.3.2.1.3 MSI-X PBA Register (0xB8; RO)

Bit(s)	Init.	Type	Description
2:0	100b	RO	PBA BIR Indicates which one of a function's BARs, located beginning at 0x10 in configuration space, is used to map the function's MSI-X PBA into memory space. BIR values: 0...5 correspond to BARs 0x10...0x24, respectively. BIR equals 4 to indicate BAR 4 and 5 are used for MSI-X PBA.
31:3	0x400	RO	PBA Offset Used as an offset from the address contained by one of the function's BARs to point to the base of the MSI-X PBA. The lower three <i>PBA BIR</i> bits are masked off (set to zero) by software to form a 32-bit QWord-aligned offset. This value is changed by hardware to be half of the value programmed to the IOV System Page Size register.

9.3.2.2 PCIe Capability Registers

The Device Control and Device Status registers have some fields which are specific per VF.

9.3.2.2.1 VF Device Control Register (0x78; RW)

Bit(s)	Init.	Type	Description
0	0b	RO	Correctable Error Reporting Enable Zero for VFs.
1	0b	RO	Non-Fatal Error Reporting Enable Zero for VFs.
2	0b	RO	Fatal Error Reporting Enable Zero for VFs.
3	0b	RO	Unsupported Request Reporting Enable Zero for VFs.
4	0b	RO	Enable Relaxed Ordering Zero for VFs.
7:5	000b	RO	Max Payload Size Zero for VFs.
9:8	00b	RO	Reserved.
10	0b	RO	Auxiliary Power PM Enable Zero for VFs.
11	0b	RO	Enable No Snoop Zero for VFs.

Bit(s)	Init.	Type	Description
14:12	000b	RO	Max Read Request Size Zero for VFs.
15	0b	RW	Initiate Function Level Reset Specific to each VF.

9.3.2.2.2 VF Device Status Register (0x7A; RW1C)

Bit(s)	Init.	Type	Description
0	0b	RW1C	Correctable Detected Indicates status of correctable error detection. Zero for VF.
1	0b	RW1C	Non-Fatal Error Detected Indicates status of non-fatal error detection. Zero for VF.
2	0b	RW1C	Fatal Error Detected Indicates status of fatal error detection. Zero for VF.
3	0b	RW1C	Unsupported Request Detected Indicates that the E610 received an unsupported request. This field is identical in all functions. The E610 cannot distinguish which function caused an error. Zero for VF.
4	0b	RO	Aux Power Detected Zero for VFs.
5	0b	RO	Transaction Pending Specific per VF. When set, indicates that a particular function (PF or VF) has issued non-posted requests that have not been completed. A function reports this bit cleared only when all completions for any outstanding non-posted requests have been received.
15:6	0x0	RO	Reserved.

9.3.3 PCIe Extended Capabilities

9.3.3.1 AER Registers

The following registers in the AER capability have a different behavior in a VF function.

Unlike the PF AER registers, these registers are not sticky since the VF is reset on FLR and on in-band reset.

9.3.3.1.1 Uncorrectable Error Status Register (0x104; RW1C)

Bit(s)	Init.	Type	Description
3:0	0x0	RO	Reserved.
4	0b	RO	Data Link Protocol Error Status
5	0b	RO	Surprise Down Error Status (Optional)
11:6	0x0	RO	Reserved.
12	0b	RW1C	Poisoned TLP Status
13	0b	RO	Flow Control Protocol Error Status
14	0b	RW1C	Completion Timeout Status
15	0b	RW1C	Completer Abort Status
16	0b	RW1C	Unexpected Completion Status

Bit(s)	Init.	Type	Description
17	0b	RO	Receiver Overflow Status
18	0b	RO	Malformed TLP Status
19	0b	RO	ECRC Error Status
20	0b	RW1C	Unsupported Request Error Status When caused by a function that claims a TLP.
21	0b	RO	ACS Violation Status
31:22	0x0	RO	Reserved.

9.3.3.1.2 Correctable Error Status Register (0x110; RW1C)

The Correctable Error Status register reports error status of individual correctable error sources on a PCIe device. When an individual error status bit is set to 1b it indicates that a particular error occurred; software can clear an error status by writing a 1b to the respective bit.

Bit(s)	Init.	Type	Description
0	0b	RW1C	Receiver Error Status
5:1	0x0	RO	Reserved.
6	0b	RW1C	Bad TLP Status
7	0b	RW1C	Bad DLLP Status
8	0b	RW1C	REPLAY_NUM Rollover Status
11:9	000b	RO	Reserved.
12	0b	RW1C	Replay Timer Timeout Status
13	0b	RW1C	Advisory Non-Fatal Error Status
31:14	0x0	RO	Reserved.

NOTE: *This page intentionally left blank.*

Chapter 10 System Manageability

Network management is an important requirement in today's networked computer environment. Software-based management applications provide the ability to administer systems while the operating system is functioning in a normal power state (not in a pre-boot state or powered-down state). The Intel® Out of Band Management fills the management void that exists when the operating system is not running or fully functional. This is accomplished by providing mechanisms by which manageability network traffic can be routed to and from a Management Controller (MC).

This section describes the supported management interfaces and hardware configurations for platform system management. It describes the interfaces to an external MC, the partitioning of platform manageability among system components, and the functionality provided by the E610 in each platform configuration.

10.1 Features

Table 10-1 lists the manageability features in the LAN controller.

Table 10-1. System Manageability Features

Description
Sideband Interfaces for connection to an external BMC • SMBus operating at up to 400 Kb/s. Using only MCTP messages. • DMTF-compliant NC-SI Interface 1.1 at 100 Mb/s • PCIe (using MCTP VDMs)
Sophisticated filters to select received packet flows for delivery or mirroring to the BMC. Each of the following filters is instantiated per-Ethernet port: • 4 MAC filters • 8 VLAN filters • 4 EtherType filters • ScPv4 and 4 IPv6 filters • 16 UDP/TCP port filters • 1 Flexible Total Cost of Ownership (TCO) filter • Address Resolution Protocol (ARP) filtering • Neighbor discovery filtering • Remote Management Control Protocol (RMCP) filtering • Internet Control Message Protocol (ICMP) filtering
Ability to internally switch packets for communication between an OS and BMC.
Supported Protocols: • DSP0222 1.1.1 - NC-SI Including ability to internally switch packets for communication between an OS and BMC. • DSP0261 1.2.0 - NC-SI over MCTP • DSP0236 1.3.1, DSP0238 1.2.0, DSP0237 1.2.0, DSP0239 1.9.0- MCTP over PCIe VDM and over SMBus • DSP0240 1.1.0 - PLDM Base • DSP0248 1.1.0, DSP0249 1.0.0- PLDM Monitoring and Control • DSP0267 1.0.1 - PLDM Firmware Update • DSP0274 1.1.1b, DSP0275 1.0.0 - SPDM (Security Protocol and Data Model Specification)

10.2 Pass-Through (PT) Functionality

PT is the term used when referring to the process of sending and receiving Ethernet traffic over the sideband interface. The E610 has the ability to route Ethernet traffic to the host operating system as well as the ability to send Ethernet traffic over the sideband interface to an external MC. See [Figure 10-1](#).

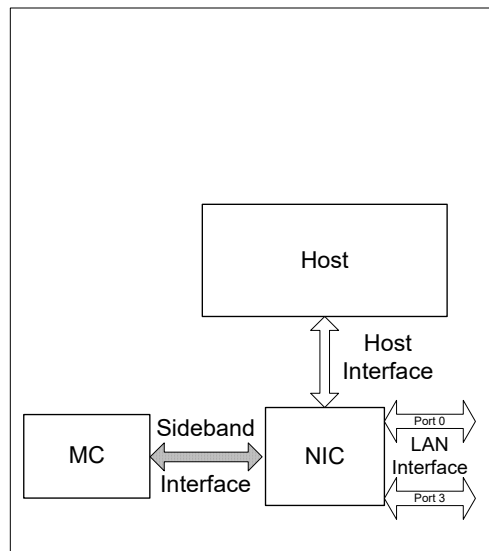


Figure 10-1. Sideband interface

The sideband interface provides a mechanism by which the E610 can be shared between the host and the MC. By providing this sideband interface, the MC can communicate with the LAN without requiring a dedicated Ethernet controller. The E610 supports three sideband interfaces:

- SMBus (as part of MCTP)
- NC-SI
- PCIe (together with MCTP) - when the system is up.

Note: The usable bandwidth for either direction is up to 400 Kb/s when using SMBus and 100 Mb/s for the NC-SI interface. When working over PCIe, the bandwidth is limited only by the PCIe bandwidth and the the E610 processing capabilities and can sustain any network bandwidth. The E610 should support MCTP over PCIe pass-through traffic at a rate of up to 1 Gb/s. The maximum packet size supported for traffic received from the LAN to the MC is 1518 bytes and an additional L2 tag. For traffic from the MC to the LAN the maximum supported packet size is 1536 bytes including all tags.

Note: In MCTP mode, the PCIe and SMBus interface can receive MCTP commands in parallel. For example, the MCTP enumeration process can be done both over SMBus and over PCIe. However, only one of the interfaces can receive NC-SI commands or pass-through traffic.

10.2.1 Supported Topologies

The E610 can support up to two connections to management controllers in some topologies. The following connections are available:

- Connection via NC-SI over RBT (RMII) (See [Section 10.6](#))
- Connection via NC-SI over MCTP. This connection can be over SMBus, PCIe or both. This connection can be used either for pass through or for control only (See [Section 10.7](#))
- Two connections — A connection via NC-SI over RBT for pass-through traffic and a connection via MCTP used only for control traffic (See [Section 10.7](#))

The channel used for pass through is defined in the *Redirection Sideband Interface* field and the channel used for control is defined in the *Control Interface* field, both in the *Common Manageability Parameters* NVM word (see [Section 6.3.49.2](#)) and are common to all the ports in the device.

10.2.2 Pass-Through Packet Routing

When an Ethernet packet reaches the E610, it is examined and compared to a number of configurable filters. These filters are configurable by the MC and include, but not limited to, filtering on:

- MAC Address
- IP Address
- UDP/IP ports
- VLAN tags
- EtherType

If the incoming packet matches any of the configured filters, it is passed to the MC. Otherwise, it is not passed.

The packet filtering process is described in [Section 10.6](#).

10.3 Components of the Sideband Interface

There are two components to a sideband interface:

- Physical layer
- Logical layer

10.3.1 Physical Layer

This is the electrical connection between the E610 and the MC.

10.3.1.1 SMBus

The SMBus physical layer is defined by the SMBus specification. The interface is made up of two connections: data and clock. Refer to the SMBus specification for details.

The SMBus can run at two speeds: 100 KHz (standard SMBus), or 400 KHz (I²C fast mode). The speed used while the E610 is the initiator of the bus is selected by the *SMBus Connection Speed* field in the SMBus Notification Timeout and Flags NVM word. When acting as a subordinate, the E610 can receive transaction with a clock running at up to 400 KHz.

10.3.1.1.1 PEC Support

SMBus transactions can be protected by using Packet Error Code (PEC). Packet error checking, when applicable, is implemented by appending a PEC byte at the end of each message transfer. The PEC byte is a CRC8 calculation on all the message bytes.

PEC is added in transmit and expected in receive for the following SMBus packets:

- ARP packets
- MCTP over SMBus transactions.

For ARA cycles, a PEC is not expected.

The following table lists the behavior of the E610 in each PEC configured mode for transactions directly handled by hardware after receiving packets with or without PEC.

Table 10-2. SMBus PEC Modes

SMBus transaction (relative to the E610)	E610 PEC Mode	Target PEC Mode ¹	
		PEC Enabled	PEC Disabled
Initiator Write ²	Enabled	(A) Target ACKs the PEC byte	(A) Target NACKs the PEC byte
	Disabled	(A) Target receives stop before expected PEC byte	(A) PEC byte is not expected
Responder Read ³	Enabled	(A) Target sends the PEC byte; PEC byte is ACKed by the responder	(A) Target does not send PEC byte and generates stop afterwards
	Disabled	(R) Target sends the PEC byte; PEC byte is NACKed by the responder	(A) Target does not send PEC byte and generates stop afterwards

1. (A) - Accept transaction; (R) - Reject transaction.

2. Used in MCTP over SMBus (transmitted transaction)s.

3. Used in MCTP over SMBus (received transactions).

Note: In both SMBus ARP and MCTP, the specification indicates that PEC must be used. However, if PEC is not used by the initiator, the transaction is still accepted and processed by the E610.

The PEC behavior is controlled by the SMBus transaction PEC bit in the SMBus Notification Timeout and Flags NVM word. If this bit is set, PEC is added for initiator SMBus write transactions. A PEC is added to responder read transactions and can be received in responder write transaction. If this bit is cleared, PEC is not added to initiator write or responder read transactions, a responder write transaction with PEC is dropped. This bit should always be set.

10.3.1.2 NC-SI over RBT

The E610 uses the DMTF standard RBT sideband interface as defined in DMTF DSP0222. This interface consists of seven lines for transmission and reception of Ethernet packets and two optional lines for arbitration among more than one physical network controller.

The RBT physical layer of NC-SI is very similar to the RMII interface, although not an exact duplicate. Refer to the NC-SI specification for details of the differences.

10.3.1.2.1 Electrical Characteristics

The E610 complies with the electrical characteristics defined in the NC-SI 1.1.0 specification.

The E610's NC-SI behavior is configured at power-up in the following manner:

- The *Multi-Drop NC-SI* NVM bit ([Section 6.3.49.3](#)) defines the NC-SI topology (point-to-point or multi-drop; the default is point-to-point).

The E610 dynamically drives its NC-SI output signals (NC-SI_DV and NC-SI_RX) as required by the sideband protocol:

- At power-up, the E610 floats the NC-SI outputs.
- If the E610 operates in point-to-point mode, it starts driving the NC-SI outputs some time following power-up.
- If the E610 operates in a multi-drop mode, it drives the NC-SI outputs as configured by the BMC.

10.3.1.3 PCIe Vendor Defined Messages (VDMs)

The E610 uses VDMs over PCIe defined in the DMTF MCTP specification to convey pass-through traffic or NC-SI control traffic. See [Section 3.1](#) for details of the PCIe interface.

10.3.2 Logical layer

10.3.2.1 SMBus

10.3.2.1.1 SMBus Transactions

This section gives a brief overview of the SMBus protocol. Following is an example for a format of a typical SMBus transaction.

Table 10-3. Typical SMBus Transaction

1	7	1	1	8	1	8	1	1
S	Responder Address	Wr	A	Command	A	PEC	A	P
	1100 001	0	0	0000 0010	0	[Data Dependent]	0	

The top row of the table identifies the bit length of the field in a decimal bit count. The middle row (bordered) identifies the name of the fields used in the transaction. The last row appears only with some transactions, and lists the value expected for the corresponding field. This value can be either hexadecimal or binary.

The SMBus controller is a initiator for some transactions and a responder for others. The differences are identified in this document.

Shorthand field names are listed in [Table 10-4](#) and are fully defined in the SMBus specification.

Table 10-4. Shorthand Field Names

Field Name	Definition
S	SMBus START Symbol.
P	SMBus STOP Symbol.
PEC	Packet Error Code.
A	ACK (Acknowledge).
N	NACK (Not Acknowledge).
Rd	Read Operation (Read Value = 1b).
Wr	Write Operation (Write Value = 0b).

10.3.2.1.2 SMBus Addressing

In MCTP mode, a single SMBus channel is exposed.

SMBus addresses (enabled from the NVM) can be re-assigned using the SMBus ARP protocol.

In addition to the SMBus address values, all parameters of the SMBus (SMBus channel selection, address mode, and address enable) can be set only through NVM configuration. Note that the NVM is read at the E610's power up and resets.

10.3.2.1.3 SMBus ARP Functionality

The E610 supports the SMBus ARP protocol as defined in the SMBus 2.0 specification. The E610 is a persistent responder address device so its SMBus address is valid after power-up and loaded from the NVM. The E610 supports all SMBus ARP commands defined in the SMBus specification both general and directed.

SMBus ARP capability can be disabled through the NVM.

10.3.2.1.3.1 SMBus ARP Flow

SMBus ARP flow is based on the status of two flags:

- Address Valid (AV): This flag is set when the E610 has a valid SMBus address.
- Address Resolved (AR): This flag is set when the E610 SMBus address is resolved (SMBus address was assigned by the SMBus ARP process).

These flags are internal E610 flags and are not exposed to external SMBus devices.

Since the E610 is a Persistent SMBus Address (PSA) device, the AV flag is always set, while the AR flag is cleared after power up until the SMBus ARP process completes. Since AV is always set, the E610 always has a valid SMBus address.

When the SMBus initiator needs to start an SMBus ARP process, it resets (in terms of ARP functionality) all devices on SMBus by issuing either Prepare to ARP or Reset Device commands. When the E610 accepts one of these commands, it clears its AR flag (if set from previous SMBus ARP process), but not its AV flag (the current SMBus address remains valid until the end of the SMBus ARP process).

Clearing the AR flag means that the E610 responds to SMBus ARP transactions that are issued by the initiator. The SMBus initiator issues a Get UDID command (general or directed) to identify the devices on the SMBus. The E610 always responds to the Directed command and to the General command only if its AR flag is not set.

After the Get UDID, The initiator assigns the E610 SMBus address by issuing an Assign Address command. The E610 checks whether the UDID matches its own UDID and if it matches, it switches its SMBus address to the address assigned by the command (byte 17). After accepting the Assign Address command, the AR flag is set and from this point (as long as the AR flag is set), the E610 does not respond to the Get UDID General command. Note that all other commands are processed even if the AR flag is set. The E610 stores the SMBus address that was assigned in the SMBus ARP process in the NVM, so at the next power up, it returns to its assigned SMBus address.

Figure 10-2 shows E610 SMBus ARP flow.

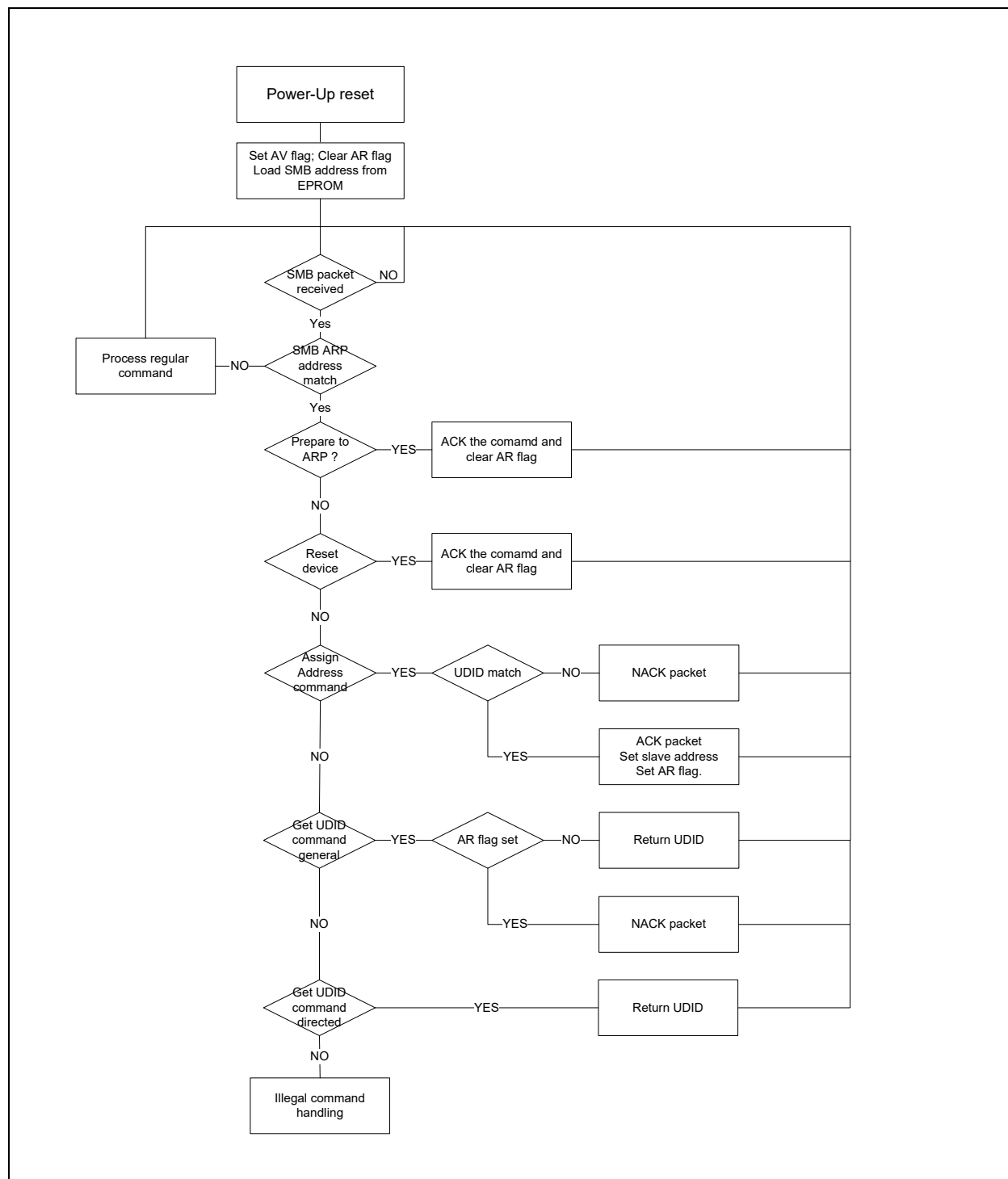


Figure 10-2. SMBus ARP Flow

10.3.2.1.3.2 SMBus ARP UDID Content

The UDID provides a mechanism to isolate each device for the purpose of address assignment. Each device has a unique identifier. The 128-bit number is comprised of the following fields:

Table 10-5. UDID

1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes	2 Bytes	2 Bytes	4 Bytes
Device Capabilities	Version/Revision	Vendor ID	Device ID	Interface	Subsystem Vendor ID	Subsystem Device ID	Vendor Specific ID
See notes that follow	See notes that follow	0x8086	0x57AC	0x0004/ 0x0024	0x0000	0x0000	See notes that follow
MSB							LSB

Where:

- Vendor ID: The device manufacturer's ID as assigned by the SBS Implementers' Forum or the PCI SIG.
Constant value: 0x8086.
- Device ID: The device ID as assigned by the device manufacturer (identified by the Vendor ID field).
Constant value: 0x57AC.
- Interface: Identifies the protocol layer interfaces supported over the SMBus connection by the device.
Bits 3:0 = 0x4 indicates SMBus Version 2.0.
Bit 5 (ASF bit) = 1 in MCTP mode.
- Subsystem Fields: These fields are not supported and return zeros.

Device Capabilities: Dynamic and Persistent Address, PEC Support bit:

7	6	5	4	3	2	1	0
Address Type		Reserved (0)	Reserved (0)	Reserved (0)	Reserved (0)	Reserved (0)	PEC Supported
0b	1b	0b	0b	0b	0b	0b	0/1b ¹
MSB							LSB

1. The value is set according to the SMBus transaction PEC bit in the NVM.

Version/Revision: UDID Version 1, Silicon Revision:

7	6	5	4	3	2	1	0
Reserved (0)	Reserved (0)	UDID Version			Silicon Revision ID		
0b	0b	001b			See the following table		
MSB							LSB

Silicon Revision ID:

Silicon Version	Revision ID
A0	000b

Vendor Specific ID: Four LSB bytes of the device Ethernet MAC Address of the relevant port. The port Ethernet address is taken from the *PRTGL_SAL* registers of the relevant ports.

1 Byte	1 Byte	1 Byte	1 Byte
MAC Address, Byte 3	MAC Address, Byte 2	MAC Address, Byte 1	MAC Address, Byte 0
MSB			LSB

10.3.2.2 NC-SI

The DMTF defines the protocol layer for the NC-SI interface. NC-SI compliant devices are required to implement a minimum set of commands. The specification also provides a mechanism for vendors to add additional capabilities through the use of OEM commands. Intel OEM NC-SI commands for the E610 are discussed in [Section 10.6.4](#). For information on base NC-SI commands, see the NC-SI specification.

NC-SI traffic can run on top of three different Physical layers:

1. NC-SI physical layer as described in [Section 10.3.1.2](#).
2. MCTP over PCIe VDM. This protocol enables control and pass-through traffic over PCIe of a NIC or a LOM device. The NC-SI over MCTP protocol is slightly different than the standard NC-SI as it includes additional NC-SI commands. This mode is usually paired with an MCTP over SMBus, where this mode is used in S0 states and the SMBus interface is used in Sx state. The MCTP protocol and the differences from standard NC-SI is described in [Section 10.7](#).
3. MCTP over SMBus. As previously described, this layer is paired with the MCTP over PCIe to support Sx modes.

The E610 exposes one NC-SI package with eight channels, one per port. The E610 implements a type C NC-SI interface (single package, common bus buffers and shared RX queue) as described in section 5.2 of the NC-SI specification.

10.3.2.2.1 Package ID Setting

The package ID can be set either from the NVM *Package ID* field in the NC-SI Configuration 1 NVM word ([Section 6.3.50.4](#)) or from an SDP pin. If set from SDP, the package ID is (0b, SDP value, 0b).

The mode used is set by the *Read NCSI Package ID from SDP* field in the NC-SI Configuration 2 NVM word ([Section 6.3.50.5](#)). The encoding is as follows:

- 00: Package ID is read from NVM Package ID field in the NC-SI Configuration 1 NVM word ([Section 6.3.50.4](#)) // Regular Card
- 10: The package ID is (NVM Package ID[2], SDP#1 value, NVM Package ID[0]). // Dell mode, OCP single SDP
- 01: The the package ID is (NVM Package ID[2], SDP#1 value, SDP#0 value,). // OCP, 2 SDPs.
- 11: Reserved

Note that when the package ID is set from the SDP pins, the used SDP should be set as an input in the relevant *GLGEN_GPIO_CTL* register. The SDPs to use is defined in the *PackageID SDP* field in the NC-SI Configuration 2 NVM word ([Section 6.3.50.5](#)).

10.3.2.2.2 Channel ID Mapping

The mapping of the channels to physical ports is according to the NC-SI Configuration 1 NVM word (Section 6.3.50.4) if the *NC-SI Channel to Port Mapping Table Valid* bit is set. If this bit is not set, the following algorithm should be used:

```
Channel_ID = 0
NC-SI_channel[3:0] = -1 // ports not associated to channels yet.
For func = 0 to 1 { // loop on all functions
    Port_ID = PFGEN_PORTNUM[func] // Port associated with function.
    If (PRTGEN_STATUS.PORT_VALID[Port_ID] && NC-SI_channel[Port_ID] == -1 && PHY
module is present) { .
        // Port is valid and port is not already associated to a channel
        NC-SI_channel[Port_ID] = Channel_ID; // assign channel
        PRT2MDEF[Port_ID] = Channel_ID;
        Channel_ID++; // go to next channel
    }
}
```

This algorithm maps channel numbers that match the order of the PCI function numbers.

Note: If firmware detects a change in the number of ports enabled, it:

- Waits until the Veto bit of all ports is cleared.
- Updates the number of exposed channel.
- Moves the package to "Initialization Required" state.

The BMC will then re-enumerate the package and discover the new number of available channels.

10.4 Packet Filtering

Since both the host operating system and BMC use the E610 to send and receive Ethernet traffic, there needs to be a mechanism by which incoming Ethernet packets can be identified as those that should be sent to the BMC rather than the host operating system.

There are two different types of filtering available. The first is filtering based upon the MAC Address. With this filtering, the BMC has at least one dedicated MAC Address and incoming Ethernet traffic with the matching MAC Address(es) are passed to the BMC. This is the simplest filtering mechanism to utilize and it allows an BMC to receive all types traffic (including, but not limited to, IPMI, NFS, HTTP etc).

The other mechanism available utilizes a highly configurable mechanism by which packets can be filtered using a wide range of parameters. Using this method, the BMC can share a MAC Address (and IP Address, if desired) with the host OS and receive only specific Ethernet traffic. This method is useful if the BMC is only interested in specific traffic, such as IPMI packets.

10.4.1 Manageability Receive Filtering

This section describes the manageability receive packet filtering flow. Packet reception by the E610 can generate one of the following results:

- Discarded.
- Sent to host memory.
- Sent to the external BMC.
- Sent to both the BMC and host memory.

The decisions regarding forwarding of packets to the host and to the BMC are separate and are configured through two sets of registers. However, the BMC may define some types of traffic as exclusive. This traffic is forwarded only to the BMC, even if it passes the filtering process of the host. These types of traffic are defined using the MNGONLY register ([Section 8.2.2.18.1](#)).

An example of packets that might be necessary to send exclusively to the BMC might be specific TCP/UDP ports of a shared MAC Address or a MAC Address dedicated to the BMC. If the BMC configures the manageability filters to send these ports to the BMC, it should configure the settings to not send them to the host, otherwise, these ports are received and handled by the host operating system.

The BMC controls the types of packets that it receives by programming receive manageability filters. The following filters are accessible to the BMC:

Table 10-6. Filters Accessible to BMC

Filters	Functionality	When Reset
Filters Enable	General configuration of the manageability filters	LAN_PWR_GOOD
Manageability Only	Enables routing of packets exclusively to the manageability.	LAN_PWR_GOOD
Manageability Decision Filters [7:0]	Configuration of manageability decision filters	LAN_PWR_GOOD
MAC Address [3:0]	Four unicast MAC manageability addresses	LAN_PWR_GOOD
VLAN Filters [7:0]	Eight VLAN tag values	LAN_PWR_GOOD
UDP/TCP Port Filters [15:0]	16 destination port values	LAN_PWR_GOOD
Flexible 128 bytes TCO Filter	Length and values for one flex TCO filter	LAN_PWR_GOOD
IPv4 and IPv6 Address Filters [3:0]	IP Address for manageability filtering	LAN_PWR_GOOD

All filtering capabilities are available on both the NC-SI and legacy SMBus interfaces. However, in NC-SI mode, to program part of the capabilities, the Intel OEM commands described in [Section 10.6.4](#) should be used.

All filters are reset only on Internal Power On Reset. Register filters that enable filters or functionality are also reset by firmware reset in NC-SI mode. These registers can be loaded from the NVM following a reset in SMBus mode. See [Section 6.3.9](#) - [Section 6.3.11](#) for description of their location in the NVM map.

The high-level structure of manageability filtering is done using two steps.

1. The packet is parsed and fields in the header are compared to programmed filters.
2. A set of decision filters are applied to the result of the first step.

Some general rules apply:

- Fragmented packets are passed to manageability but not parsed beyond the IP header.
- Packets with L2 errors (CRC, alignment, etc.) are not forwarded to the BMC.
- Packets longer than 2KB are filtered out.
- The filtering relates only to the outer L3/L4 header. In tunneled packets, the inner L2, L3 and L4 header parameters are not considered for manageability filtering.

The following sections describe the manageability filtering, followed by the final filtering rules. The filtering rules are created by programming the decision filters as described in [Section 10.4.4](#).

10.4.2 L2 Filters

10.4.2.1 MAC and VLAN Filters

The manageability MAC filters allow comparison of the Destination MAC Address to one of 4 filters defined in the MMAH and MMAL registers.

The VLAN filters allow comparison of the 12-bit VLAN tag to one of 8 filters defined in the MAVTV registers.

10.4.2.2 EtherType Filters

Manageability L2 EtherType filters allow filtering of received packets based on the Layer 2 EtherType field. The L2 type field of incoming packets is compared against the EtherType filters programmed in the Manageability EtherType Filter (METF; up to 4 filters); the result is incorporated into decision filters.

Each Manageability EtherType filter can be configured as pass (positive) or reject (negative) using a polarity bit. For the reverse polarity mode to be effective and block certain type of packets, the EtherType filter should be part of all the enabled decision filters.

An examples for usage of L2 EtherType filters is to determine the destination of 802.1X control packets. The 802.1X protocol is executed at different times in either the management controller or by the Host. L2 EtherType filters are used to route these packets to the proper agent.

In addition to the flexible EtherType filters, the E610 supports 2 fixed EtherType filters used to block NC-SI control traffic (0x88F8) and flow control traffic (0x8808) from reaching the manageability interface. The NC-SI EtherType is used for communication between the management controller on the NC-SI link and the E610. Packets coming from the network are not expected to carry this EtherType and such packets are blocked to prevent attacks on the management controller. Flow control packets should be consumed by the MAC and as such are not expected to be forwarded to the management interface.

10.4.3 L3/L4 Filtering

The manageability filtering stage combines checks done at previous stages with additional L3/L4 checks to make a the decision on whether to route a packet to the BMC. The following sections describe the manageability filtering done at layers L3/L4 and final filtering rules.

Note: The L3 filters do not match tunneled fields (L3, L4 or ARP).

10.4.3.1 ARP Filtering

ARP filtering — The E610 supports filtering of ARP request packets (initiated externally) and ARP responses (to requests initiated by the BMC).

To limit the reception of ARP packets to the ARP packets dedicated to this station (ARP target IP = BMC IP), the ARP request/response filter can be bind to specific IP Address, by setting both the ARP Request/Response and the IP AND bits in an MDEF filter, as the *IP* bit is set also if there is a match on the target IP (the TPA field in the ARP packet) of an ARP request or an ARP response.

Note: If the OR section of the MDEF is all cleared and one of the IPv4 address are set, ARP packets matching the IP Address pass the filter. If these packets should be dropped, an OR EtherType filter with a a value of 0x0800 (IPv4) should be added.

10.4.3.2 Neighbor Discovery Filtering and MLD

The E610 supports filtering of the following ICMPv6 packets.

Neighbor Discovery packets:

- 0x86 (134d) - Router Advertisement.
- 0x87 (135d) - Neighbor Solicitation.
- 0x88 (136d) - Neighbor Advertisement.
- 0x89 (137d) - Redirect.

MLD packets:

- 0x82 (130d) - MLD Query
- 0x83 (131d) - MLDv1 Report
- 0x84 (132d) - MLD Done
- 0x8F (143d) - MLDv2 Report

The Neighbor Discovery packets has dedicated enables for each type in the decision filters. For MLD, a single enable controls the forwarding of all the MLD packets. This means that either all the MLD packets types are selected for reception or none of them.

10.4.3.3 RMCP Filtering

The E610 supports filtering by fixed destination port numbers, port 0x26F and port 0x298. These ports are IANA reserved for RMCP.

In SMBus mode, there are filters that can be enabled for these ports. When using NC-SI, they are not specifically available. However, the general filtering mechanism can be utilized to filter incoming ARP traffic.

10.4.3.4 Flexible Port Filtering

The E610 implements 16 flex destination port filters. The E610 directs packets whose L4 destination port matches to the BMC. The BMC must ensure that only valid entries are enabled in the decision filters.

10.4.3.5 IP Address Filtering

The E610 supports filtering by destination IP Address using IPv4 and IPv6 address filters. These are dedicated to manageability. The E610 provides four IPv6 address filters or three IPv6 addresses and four IPv4 address filters.

The IPv4 match rises also for ARP packets for which the Target IP matches the IP Address in the MIPAF4 register.

10.4.3.6 Checksum Filtering

If bit `MANC.EN_XSUM_FILTER` is set, the E610 directs packets to the BMC only if they match all other filters previously described as well as pass L3/L4 checksum (if it exists).

The E610 may be instructed to directs packets to the BMC only if they pass L3/L4 checksum (if they exist) in addition to matching other filters previously described.

To enable the XSUM filtering when using NC-SI, use the Enable Checksum Offloading command (see [Section 10.6.4.13](#)).

10.4.4 Flexible 128 Byte Filter

The E610 provides one flex TCO filter. This filter looks for a pattern match within the first 128 bytes of the packet.

Flex filters are temporarily disabled when read from or written to by the host. Any packet received during a read or write operation is dropped. Filter operation resumes once the read or write access completes.

10.4.4.1 Flexible Filter Structure

The filter is composed of the following fields:

- **Flexible Filter Length** — This field indicates the number of bytes in the packet header that should be inspected. The field also indicates the minimal length of packets inspected by the filter. Packet below that length are not inspected. Valid values for this field are: $8*n$, where $n=1...16$.
- **Data** — This is a set of up to 128 bytes comprised of values that header bytes of packets are tested against.
- **Mask** — This is a set of 128 bits corresponding to the 128 data bytes that indicate for each corresponding byte if is tested against its corresponding byte. The general filter is 128 bytes that the BMC configures; all of these bytes may not be needed or used for the filtering, so the mask is used to indicate which of the 128 bytes are used for the filter.

Each filter tests the first 128 bytes (or less) of a packet, where not all bytes must necessarily be tested.

10.4.4.2 TCO Filter Programming

Programming each filter is done using the following commands (NC-SI or SMBus) in a sequential manner:

1. **Filter Mask and Length** — This command configures the following fields:
 - a. **Mask** — A set of 16 bytes containing the 128 bits of the mask. Bit 0 of the first byte corresponds to the first byte on the wire.
 - b. **Length** — A 1-byte field indicating the length.
2. **Filter Data** — The filter data is divided into groups of bytes. as described below:

Group	Test Bytes
0x0	0-29
0x1	30-59
0x2	60-89
0x3	90-119
0x4	120-127

Each group of bytes need to be configured using a separate command, where the group number is given as a parameter. The command has the following parameters:

- Group number — A 1-byte field indicating the current group addressed.
- Data bytes — Up to 30 bytes of test-bytes for the current group.

10.4.5 Configuring Manageability Filters

There are a number of pre-defined filters that are available for the BMC to enable, such as ARPs and IPMI ports 298h 26Fh. These are generally enabled by setting the appropriate bit within the MANC register using specific commands.

For more advanced filtering needs, the BMC has the ability to configure a number of configurable filters. It is a two-step process to use these filters. They must first be configured and then enabled.

10.4.5.1 Manageability Decision Filters

Manageability decision filters (MDEF) are a set of eight filters, each with the same structure. The filtering rule for each decision filter is programmed by the BMC and defines which of the L2, VLAN, EtherType and L2/L3 filters participate in decision making. Any packet that passes at least one rule is directed to manageability and possibly to the host.

The inputs to each decision filter are:

- Packet passed a valid management L2 exact address filter.
- Packet is a broadcast packet.
- Packet has a VLAN header and it passed a valid manageability VLAN filter.
- Packet matched one of the valid IPv4 or IPv6 manageability address filters.
- Packet is a multicast packet.

- Packet passed ARP filtering (request or response).
- Packet passed neighbor solicitation filtering.
- Packet passed MLD filtering.
- Packet passed 0x298/0x26F port filter.
- Packet passed a valid flex port filter.
- Packet passed a valid flex TCO filter.
- Packet passed or failed an L2 EtherType filter.
- Packet passed or failed Flow Control or NC-SI L2 EtherType Discard filter.

The structure of each decision filter is shown in [Figure 10-3](#). A boxed number indicates that the input is conditioned by a mask bit defined in the MDEF register and MDEF_EXT register for this rule. Decision filter rules are as follows:

- At least one bit must be set in a register. If all bits are cleared (MDEF/MDEF_EXT = 0x0000), the decision filter is disabled and ignored.
- All enabled AND filters must match for the decision filter to match. An AND filter not enabled in the MDEF/MDEF_EXT registers is ignored. If an AND filter is preceded by a OR filter, at least one of the enabled OR inputs must match for the filter to pass.
- If no OR filter is enabled in the register, the OR filters are ignored in the decision (the filter might still match).
- If one or more OR filters are enabled in the register, at least one of the enabled OR filters must match for the decision filter to match.

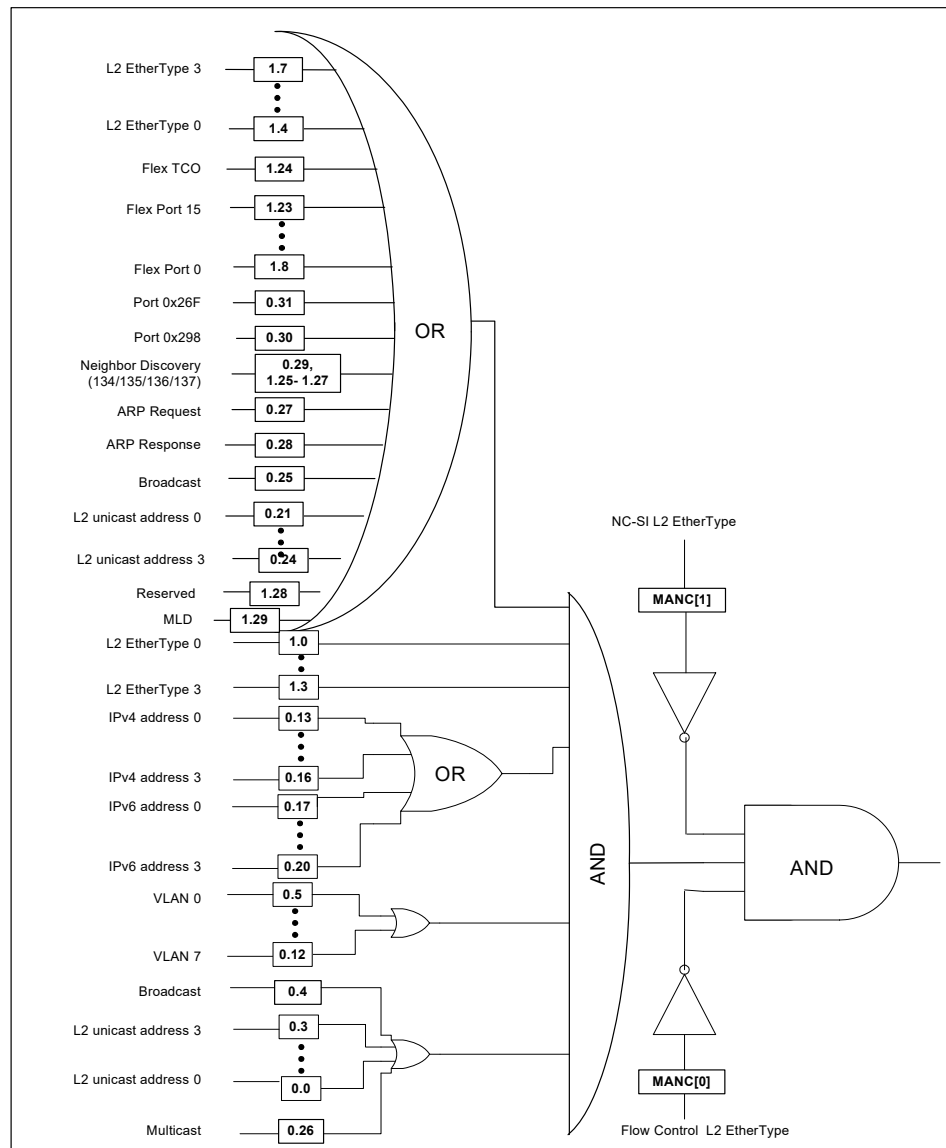


Figure 10-3. Manageability Decision Filters

A decision filter (for any of the 8 filters) defines which of the above inputs is enabled as part of a filtering rule. The BMC programs two 32-bit registers per rule (MDEF[7:0] & MDEF_EXT[7:0]) with the settings as described in [Section 8.2.2.18.6](#) and [Section 8.2.2.18.7](#). A set bit enables its corresponding filter to participate in the filtering decision.

In addition to the controls described above, the MDEF_EXT.APPLY_TO_HOST_TRAFFIC and MDEF_EXT.APPLY_TO_NETWORK_TRAFFIC bits defines which traffic is compared to this filter. At least one of these bits must be set for the filter to be valid.

If the MDEF_EXT.APPLY_TO_HOST_TRAFFIC bit is set, the traffic from the host is candidate for this filter. If the MDEF_EXT.APPLY_TO_NETWORK_TRAFFIC bit is set, the traffic from the network is candidate for this filter. If both bits are set, this filter is applied to all traffic.

10.4.5.2 Exclusive Traffic

The decisions regarding forwarding of packets to the host for LAN traffic or to the LAN for host traffic are independent from the management decision filters. However, the BMC may define some types of traffic as exclusive. The behavior for such traffic is defined by the using the bits corresponding to the decision filter in the MNGONLY register (one bit per each of the eight decision rules) and the MDEF_EXT.APPLY_TO_HOST_TRAFFIC and MDEF_EXT.APPLY_TO_NETWORK_TRAFFIC bits.

Table 10-7 describes the behavior in each case. If one or more filters match the traffic and at least one of the filters is set as exclusive, the traffic is treated as exclusive.

Table 10-7. Exclusive Traffic Behavior

Traffic Source	Filter Matches		Filter Does Not Match
	MNGONLY = 0	MNGONLY = 1	N/A
From network	Traffic is forwarded to the manageability. Traffic is forwarded to the host according to host filtering	Traffic is forwarded only to manageability.	Traffic is forwarded to the host according to host filtering
From host	Traffic is forwarded to the manageability and to the LAN	Traffic is forwarded only to manageability.	Traffic is forwarded to the LAN

Any traffic matching any of the configurable filters (see [Section 10.4.5.1](#)) can be used as filters to pass traffic to the host.

Table 10-8. MNGONLY Register Description and Usage

Bits	Description	Default
0	Decision Filter 0	Determines if packets that have passed decision filter 0 are sent exclusively to the manageability path.
1	Decision Filter 1	Determines if packets that have passed decision filter 1 are sent exclusively to the manageability path.
2	Decision Filter 2	Determines if packets that have passed decision filter 2 are sent exclusively to the manageability path.
3	Decision Filter 3	Determines if packets that have passed decision filter 3 are sent exclusively to the manageability path.
4	Decision Filter 4	Determines if packets that have passed decision filter 4 are sent exclusively to the manageability path.
5	Unicast and Mixed	NC-SI mode: Determines if unicast and mixed packets are sent exclusively to the manageability path. SMBus mode: Determines if packets that have passed decision filter 5 are sent exclusively to the manageability path.
6	Global Multicast	NC-SI mode: Determines if multicast packets are sent exclusively to the manageability path. SMBus mode: Determines if packets that have passed decision filter 6 are sent exclusively to the manageability path.
7	Broadcast	NC-SI mode: Determines if broadcast packets are sent exclusively to the manageability path. SMBus mode: Determines if ARP packets are sent exclusively to the manageability path.
31:8	Reserved	Reserved.

The MNGONLY register is also configurable when using NC-SI using the Set Intel Filters — Manageability Only Command (see [Section 10.6.4.5.3](#)).

All manageability filters are controlled by the BMC only and not by the LAN device driver.

10.4.5.3 Global Controls

On top of the MDEF filters, the MANC register contains some global controls applied to all the packets to be candidate for manageability filtering:

- Receive Enable bits:
 - The *RCV_TCO_EN* field controls the reception of manageability traffic. It should be set only if one of the following bits is set also.
 - The *EN_BMC2OS* bit controls the reception of manageability traffic from the host.
 - The *EN_BMC2NET* bit controls the reception of manageability traffic from the network.
- VLAN filtering:

To support the NC-SI VLAN modes the following controls are provided:

 - The *FIXED_NET_TYPE* field controls if only VLAN tagged or VLAN un-tagged traffic is received. If this bit is cleared both types are received. If it is set, only the type described by the *NET_TYPE* field is accepted.
 - If set, the *NET_TYPE* field indicates that only VLAN tagged traffic is received, if cleared only packets without VLAN is accepted. This field is validated by the *FIXED_NET_TYPE* field.

Both fields relates to the inner VLAN.

10.4.6 Filtering Programming Interfaces

The E610 provides multiple options to program the forwarding filters, depending on the interface used and the level of flexibility needed. [Table 10-9](#) describes the different options and points to the description of the relevant commands.

Table 10-9. Filtering Programming Interfaces

Interface	Flexible/Abstract	Description
NC-SI (over RMII or over MCTP)	Abstract (dedicated MAC Address)	The regular NC-SI commands can be used to allow forwarding based on a dedicated MAC Address. The list of supported commands can be found in Section 10.6.2.1 . When using these commands, one of the two other modes can be used to add finer grain filtering.
	Flexible	This interface uses the packet reduction commands to reduce the forwarding scope of the filters set by the regular NC-SI commands and the packet addition commands to add new packet types to the forwarding rules.

10.4.7 Possible Configurations

This section describes ways of using management filters. Actual usage may vary.

10.4.7.1 Dedicated MAC Packet Filtering

- Select one of the eight rules for dedicated MAC filtering.
- Load Host MAC Address to one of the management MAC Address filters and set the appropriate bit in field 3:0 of the MDEF register.

- Set other bits to qualify which packets are allowed to pass through. For example:
 - Set bit 5 in MDEF to qualify with the first manageability VLAN.
 - Set relevant bits 13 to 20 in MDEF to qualify with a match to one of the IP Addresses.
 - Set any L3/L4 bits (bits 27 to 31 in MDEF and bits 16 to 23 in MDEF_EXT) to qualify with any of a set of L3/L4 filters.

10.4.7.2 Broadcast Packet Filtering

- Select one of the eight rules for broadcast filtering.
- Set bit 25 in MDEF of the decision rule to enforce broadcast filtering.
- Set other bits to qualify which broadcast packets are allowed to pass through. For example:
 - Set bit 5 in MDEF to qualify with the first manageability VLAN.
 - Set relevant bits 13 to 20 in MDEF to qualify with a match to one of the IP Addresses.
 - Set any L3/L4 bits (bits 27 to 31 in MDEF, and bits 16 to 23 in MDEF_EXT) to qualify with any of a set of L3/L4 filters.

10.4.7.3 VLAN Packet Filtering

- Select one of the eight rules for VLAN filtering.
- Set bit 5 to 12 in MDEF to qualify with the relevant manageability VLANs.
- Set other bits to qualify which VLAN packets are allowed to pass through. For example:
 - Set any L3/L4 bits (bits 27 to 31 in MDEF, and bits 16 to 23 in MDEF_EXT) to qualify with any of a set of L3/L4 filters.

10.4.7.4 IPv6 Filtering

IPv6 filtering is done using the following IPv6-specific filters:

- IP Unicast filtering — requires filtering for Link Local address and a Global address. Filtering setup might depend on whether the MAC Address is shared with the host or dedicated to manageability:
 - Dedicated MAC Address (for example, dynamic address allocation with DHCP does not support multiple IP Addresses for one MAC Address). In this case, filtering can be done at L2 using two dedicated unicast MAC filters.
 - Shared MAC Address (for example, static address allocation sharing addresses with host). In this case, filtering needs to be done at L3, requiring two IPv6 address filters, one per address.
- A neighbor Discovery filter — The E610 supports IPv6 neighbor Discovery protocol. Since the protocol relies on multicast packets, the E610 supports filtering of these packets. IPv6 multicast addresses are translated into corresponding Ethernet multicast addresses in the form of 33-33-xx-xx-xx-xx, where the last 32 bits of address are taken from the last 32 bits of the IPv6 multicast address. As a result, two direct MAC filters can be used to filter IPv6 solicited-node multicast packets as well as IPv6 all node multicast packets.

10.4.7.5 Receive Filtering with Shared IP

When using the Legacy SMBus interface, it is possible to share the host MAC and IP Address with the BMC. This functionality is also available when using base NC-SI using Intel OEM commands.

When the BMC shares the MAC and IP Address with the host, receive filtering is based on identifying specific flows through port allocation. The following setting might be used when using the legacy SMBus interface:

- Select one of the eight rules.
- Set a manageability dedicated MAC filter to the host MAC Address and set the matching bit (0-3) in the MDEF register.
- If VLAN is used for management, load one or more management VLAN filters and set the matching bit (5-12) in the MDEF register.

ARP filter/neighbor Discovery filter is enabled when the BMC is responsible for handling the ARP protocol. Set bit 27 or bit 28 in the MDEF register for this functionality.

10.4.8 Determining Manageability MAC Address

If the BMC wishes to use a dedicated MAC Address or configure the automatic ARP response mechanism (only available in SMBus mode), it may be beneficial for the BMC to be able to determine the MAC Address used by the host.

Both the NC-SI and SMBus interfaces provide an Intel OEM command to read the System MAC Address.

A possible use for this is that the MAC Address programmed at manufacturing time does not increment by one each time, but rather by two. In this way, the BMC can read the System MAC Address and add one to it and be guaranteed of a unique MAC Address.

Determining the IP Address being used by the host is beyond the scope of this document.

10.5 OS-to-BMC Traffic

10.5.1 Overview

Traditionally, the communication between a host and the local BMC is not handled through the network interface and requires a dedicated interface such as an IPMI KCS interface. The E610 allows the host and the local BMC communication via the regular pass-through interface, and thus allow management of a local console using the same interface used to manage any BMC in the network.

When this flow is used, the host sends packets to the BMC through the network interface. The E610 examines these packets and then decides if they should be forwarded to the BMC. On the inverse path, when the BMC sends a packet on the pass-through interface, the E610 checks if it should be forwarded to the network, the host, or both. [Figure 10-4](#) describes the flow for OS-to-BMC traffic for the NC-SI over RMII case. OS2BMC is available also when working over MCTP. It is not supported in legacy SMBus mode.

The OS-to-BMC flow can be enabled using the *OS2BMC enable* field for the relevant port in the *OS 2 BMC configuration* structure of the NVM.

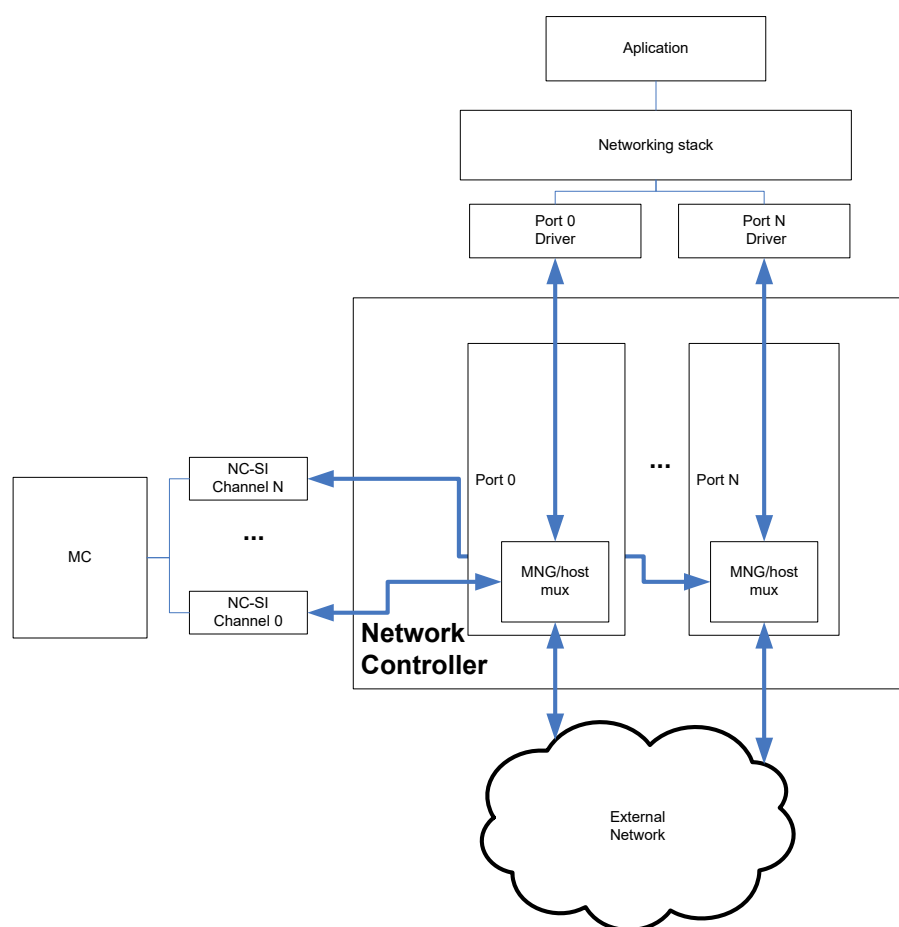


Figure 10-4. OS-to-BMC Diagram

The OS-to-BMC flow is enabled only for ports enabled by the NC-SI “Enable Channel” command or via the *OS-to-BMC Enable* field for the relevant port in the OS-to-BMC configuration structure of the NVM.

OS2BMC traffic must comply with NC-SI specifications and is therefore limited to maximum sized frames of 1536 bytes (in both directions).

10.5.2 Filtering

When OS-to-BMC traffic is enabled, the filters used for network to BMC traffic are also used for OS-to-BMC traffic. Traffic considered as exclusive to the BMC (Relevant bit in MNGONLY is set) is also considered as exclusive to the BMC when sent from the Host and not forwarded to the network.

10.5.2.1 Handling of OS-to-BMC Packets

All the regular transmit offloads are also available for OS-to-BMC packets.

10.5.2.2 BMC-to-OS Filtering

When OS-to-BMC is enabled, as with regular BMC transmit traffic, the port (OS or network) to which the packet is sent is fixed according to the source MAC Address of the packet. After that, the BMC traffic is filtered according to the L2 Host filters of the selected port (as described in [Section 7.1.2](#)). According to the results of the filtering the packet can be forwarded to the OS, the network or both.

The following rules apply to the forwarding of OS packets:

- If *BMC to net* is disabled, all the traffic from the BMC is sent to the Host.
- If *BMC to host* is disabled, all the traffic from the BMC is sent to the network.
- If both *BMC to net* and *BMC to host are enabled*, the packet is forwarded only according to the destination MAC Address and VLAN tag. Unicast packets that matches one of the exact filters (RAH/RAL) are sent only to the Host. Other packets that pass the L2 Host filtering are sent to both the Host and the network. Packets that do not pass the L2 host filtering are sent only to the network.

Note: In virtualization mode, if packet is received by host only due to default pooling (PFVTCTL.DIS_DEF_POOL bit is cleared), it is not sent to the host (even if *BMC to net* is disabled).

10.5.2.3 Queuing of Packets Received from the BMC

Packets received from the BMC are queued in the default queue.

10.5.2.4 Offloads of Packets Received from the BMC

Packets received from the BMC and forwarded to the OS do not pass the same path as regular network packets. Thus parts of the offloads provided for the network packets are not available for the BMC packets. Packet received from the BMC are identified by the *RDESC.STATUS.BMC* bit.

The following list describes which offloads are available for BMC packets:

- CRC is checked and removed on the BMC packets.
- The BMC packets are not detected as time sync packet. The *RDESC.STATUS.TS* is always clear for these packets.
- In systems where the double VLAN feature is enabled (*CTRL_EXT.EXTENDED_VLAN* is set), the *VEXT* bit is valid for BMC packets and reflects the presence of a tag with an EtherType matching the value in *EXVET* register.

Note: In systems that uses double VLAN, the BMC is expected to send all packets (apart from NC-SI commands) with the outer VLAN included. Failing to do so may cause corruptions to the packet received by the OS

- The *RDESC.ERRORS* field is always cleared for these packets.

Note: Traffic sent from the BMC does not cause a PME event, even if it matches one of the wake-up filters set by the port.

10.5.3 Blocking of Network to BMC Flow

In some systems the BMC may have its own private connection to the network and may use the E610 port only for the OS-to-BMC traffic. In this case, the BMC to network flow should be blocked while enabling the OS-to-BMC and OS to network flows.

This can be done by clearing the *MANC.EN_BMC2NET* bit for the relevant port. The BMC can control this functionality using the "Enable Network to BMC flow" and "Disable Network to BMC flow" NC-SI OEM commands. This can also be controlled using the *Network to BMC disable* field in the NVM "OS2BMC Configuration Structure".

Notes: When network to BMC flow is blocked and OS-to-BMC flow is enabled, all the traffic from the BMC is sent to the OS without any check. The OS traffic filtering is still done using the regular decision filters.

The NC-SI channel should not be enabled for receive or transmit before at least one of the *EN_BMC2NET* or *EN_BMC2OS* fields is set, unless used for AEN transmissions only. In this case, the channel may be enabled for receive, but all receive filters should be cleared.

10.5.4 OS2BMC and Flow Control

The traffic between the host and manageability uses the same buffers as any loopback traffic. Thus it flows through the transmit buffer and then through the receive buffer. If the transmit buffer is flow controlled, the host to BMC traffic is also stopped. If the receive buffer is full, the traffic is dropped or the transmit is stopped according to the flow control policy of this traffic class.

Packets received to the manageability (either from host or from network) may be dropped if the manageability internal buffers are full.

10.5.5 Statistics

Packets sent from the OS to the BMC should be counted by all statistical counters as packets sent by the OS. If they are sent to both the network and to the BMC, they are counted once.

Packets sent from the BMC to the host are counted as packets received by the host. If they are sent to the host and to the network, they are counted both as received packets and as packet transmitted to the network.

In addition, the E610 supports the following statistical counters that measure just the BMC-to-OS and OS-to-BMC traffic:

- O2BGPTC: OS-to-BMC packets received by BMC
- O2BSPC: OS-to-BMC packets transmitted by OS and received by manageability buffer.
- B2OSPC: BMC-to-OS packets sent by BMC
- B2OGPRC: BMC-to-OS packets received by OS.

The driver can use these statistics to count packets dropped by the E610 during the transfer between the OS and the BMC or the LAN and the BMC as follows:

- Dropped packets in BMC-to-OS path = $B2OSPC - B2OGPRC$
- Dropped packet in BMC to LAN path = $B2OSDPC - (B2OSPC - B2OGPRC)$
- Dropped packets in OS-to-BMC path = $O2BSPC - O2BGPTC$
- Dropped packets in LAN to BMC path = $MNGPDC$

See [Section 7.1.7.2](#) and [Section 7.1.5.2](#) for details of the statistics hierarchy.

10.5.6 OS-to-BMC Enablement

The E610 supports the unified network software model for OS-to-BMC traffic, where the OS-to-BMC traffic is shared with the regular traffic. In this model, there is no need for a special configuration of the OS networking stack or the BMC stack. However, if the link is down, the OS-to-BMC communication is stopped.

To enable OS-to-BMC, either:

- Enable OS2BMC in the *Port Traffic Type* field in the *Traffic Type Parameters* NVM word for the relevant port.
- Send an Enable Network to BMC Command

Note: When OS2BMC is enabled, OS must avoid sending packets longer than 1.5 KB to BMC.

10.6 Network Controller Sideband Interface (NC-SI) PT interface

The NC-SI is a DMTF industry standard protocol for the sideband interface. NC-SI uses a modified version of the industry standard RMII interface for the physical layer (RBT) as well as defining a new logical layer.

The NC-SI specification can be found at:

<http://www.dmtf.org/>

10.6.1 Overview

10.6.1.1 Terminology

The terminology in this section is taken from the NC-SI specification.

Table 10-10. NC-SI Terminology

Term	Definition
Frame Versus Packet	Frame is used in reference to Ethernet, whereas packet is used everywhere else.
External Network Interface	The interface of the network controller that provides connectivity to the external network infrastructure (port).
Internal Host Interface	The interface of the network controller that provides connectivity to the host OS running on the platform.
Management Controller (BMC)	An intelligent entity comprising of hardware/firmware/software, that resides within a platform and is responsible for some or all management functions associated with the platform (BMC, service processor, etc.).
Network Controller (NC)	The component within a system that is responsible for providing connectivity to the external Ethernet network world.
Remote Media	The capability to allow remote media devices to appear as if they were attached locally to the host.
Network Controller Sideband Interface	The interface of the network controller that provides connectivity to a management controller. It can be shorten to sideband interface as appropriate in the context.
Interface	This refers to the entire physical interface, such as both the transmit and receive interface between the management controller and the network controller.
Integrated Controller	The term integrated controller refers to a network controller device that supports two or more channels for NC-SI that share a common NC-SI physical interface. For example, a network controller that has two or more physical network ports and a single NC-SI bus connection.
Multi-Drop	Multi-drop commonly refers to the case where multiple physical communication devices share an electrically common bus and a single device acts as the initiator of the bus and communicates with multiple responder or target devices. In NC-SI, a management controller serves the role as the initiator, and the network controllers are the target devices.
Point-to-Point	Point-to-point commonly refers to the case where only two physical communication devices are interconnected via a physical communication medium. The devices might be in a initiator/responder relationship, or could be peers. In NC-SI, point-to-point operation refers to the situation where only a single management controller and single network controller package are used on the bus in a initiator/responder relationship where the management controller is the initiator.
Channel	The control logic and data paths supporting NC-SI pass-through operation on a single network interface (port). A network controller that has multiple network interface ports can support an equivalent number of NC-SI channels.

Table 10-10. NC-SI Terminology [continued]

Term	Definition
Package	One or more NC-SI channels in a network controller that share a common set of electrical buffers and common buffer control for the NC-SI bus. Typically, there will be a single, logical NC-SI package for a single physical network controller package (chip or module). However, the specification allows a single physical chip or module to hold multiple NC-SI logical packages.
Control Traffic/Messages/Packets	Command, response and notification packets transmitted between BMC and the E610 for the purpose of managing NC-SI.
Pass-Through Traffic/Messages/Packets	Non-control packets passed between the external network and the BMC through the E610.
Channel Arbitration	Refer to operations where more than one of the network controller channels can be enabled to transmit pass-through packets to the BMC at the same time, where arbitration of access to the RXD, CRS_DV, and RX_ER signal lines is accomplished either by software or hardware means.
Logically Enabled/Disabled NC	Refers to the state of the network controller wherein pass-through traffic is able/unable to flow through the sideband interface to and from the management controller, as a result of issuing Enable/Disable Channel command.
NC RX	Defined as the direction of ingress traffic on the external network controller interface
NC TX	Defined as the direction of egress traffic on the external network controller interface
NC-SI RX	Defined as the direction of ingress traffic on the sideband enhanced NC-SI Interface with respect to the network controller.
NC-SI TX	Defined as the direction of egress traffic on the sideband enhanced NC-SI Interface with respect to the network controller.

10.6.1.2 System Topology

In NC-SI each physical endpoint (NC package) can have several logical responders (NC channels).

NC-SI defines that one MC and up to four network controller packages can be connected to the same NC-SI link.

Figure 10-5 shows an example topology for a single MC (also know as BMC) and a single NC package. In this example, the NC package has two NC channels.

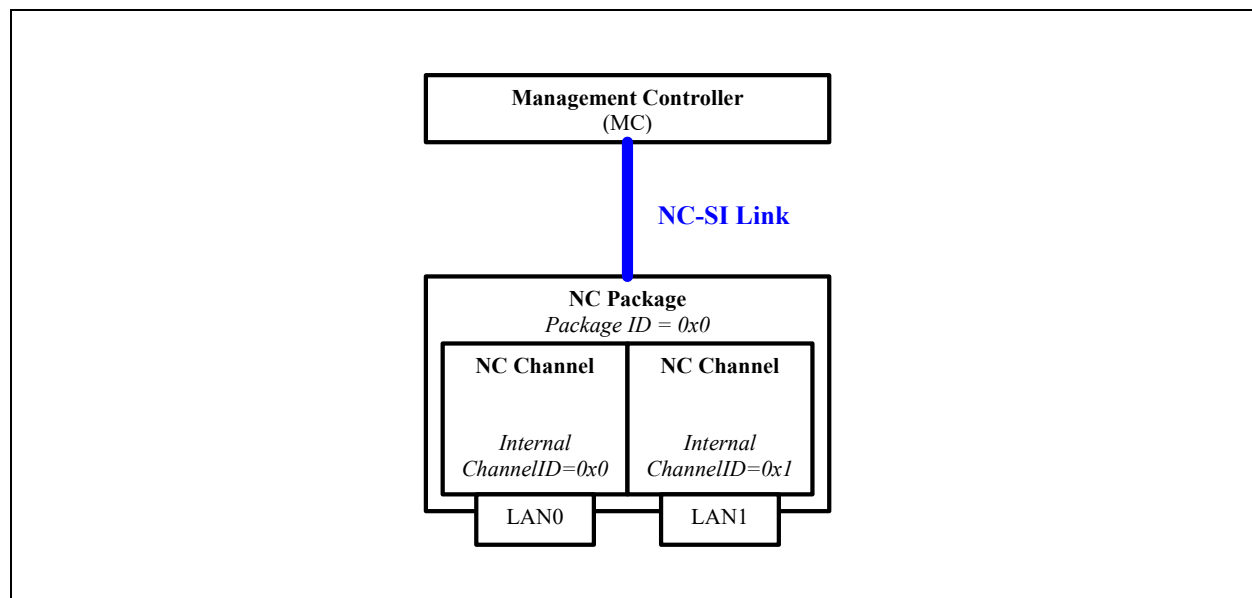


Figure 10-5. Single NC package, Two NC Channels

Figure 10-6 shows an example topology for a single MC and two NC packages. In this example, one NC package has two NC channels and the other has only one NC channel. Scenarios in which the NC-SI lines are shared by multiple NCs (Figure 10-6) mandate an arbitration mechanism.

The arbitration mechanism is described in Section 10.6.9.1.

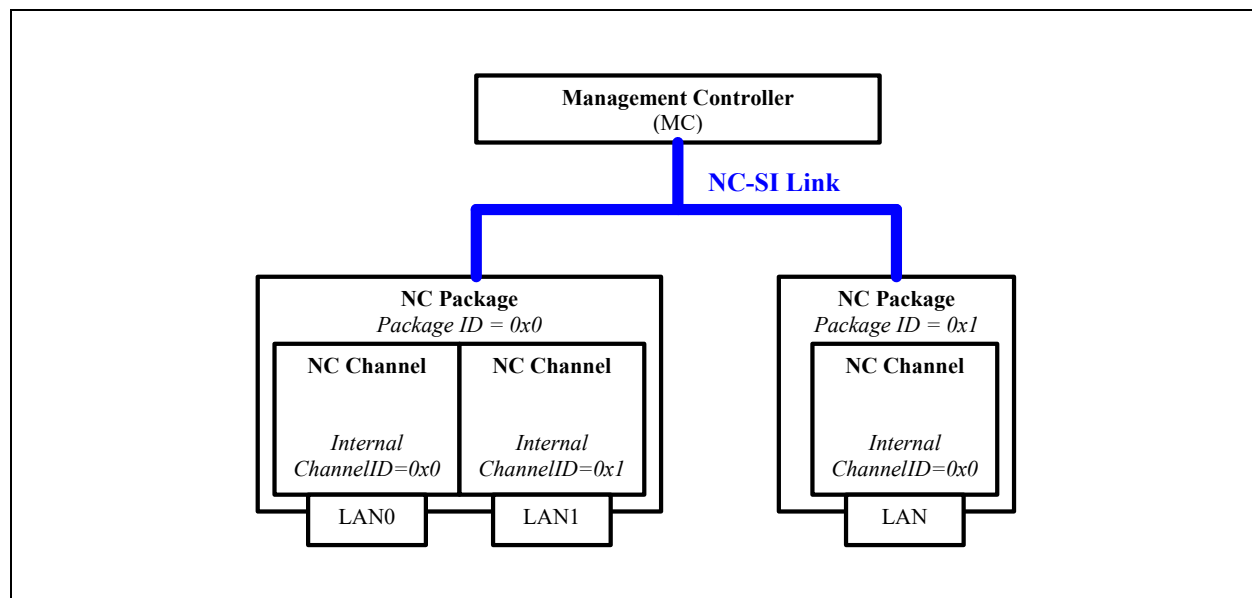


Figure 10-6. Two NC Packages (Left, with Two NC Channels and Right, with One NC Channel)

Note: Channel numbers should match PCI function numbers. If more than one function is defined on a port, the function with the lowest value associated with this port is used. See Section 10.3.2.2.2 for details.

10.6.1.3 Data Transport

Since NC-SI is based upon the RMII transport layer, data is transferred in the form of Ethernet frames. NC-SI defines two types of transmitted frames:

1. Control frames:
 - a. Configures and control the interface.
 - b. Identified by a unique EtherType in their L2 header.
2. PT frames:
 - a. Actual LAN pass-through frames transferred from/to the MC.
 - b. Identified as not being a control frame.
 - c. Attributed to a specific NC channel by their source MAC Address (as configured in the NC by the MC).

10.6.1.3.1 Control Frames

NC-SI control frames are identified by a unique NC-SI EtherType (0x88F8).

Control frames are used in a single-threaded operation, meaning commands are generated only by the MC and can only be sent one at a time. Each command from the MC is followed by a single response from the NC (command-response flow), after which the MC is allowed to send a new command.

The only exception to the command-response flow is the Asynchronous Event Notification (AEN). These control frames are sent unsolicited from the NC to the MC.

AEN functionality by the NC must be disabled by default, until activated by the MC using the AEN Enable commands.

In order to be considered a valid command, a control frame must:

1. Comply with the NC-SI header format.
2. Be targeted to a valid channel in the package via the *Package ID* and *Channel ID* fields. For example, to target a NC channel with package ID of 0x2 and internal channel ID of 0x5, the MC must set the channel ID inside the control frame to 0x45. The channel ID is composed of three bits of package ID and five bits of internal channel ID.
3. Contain a correct payload checksum (if used).
4. Meet any other condition defined by NC-SI.

There are also commands (such as select package) targeted to the package as a whole. These commands must use an internal channel ID of 0x1F.

For details, refer to the NC-SI specification.

10.6.1.3.2 NC-SI Frames Receive Flow

Figure 10-7 shows the flow for frames received on the NC from the MC.

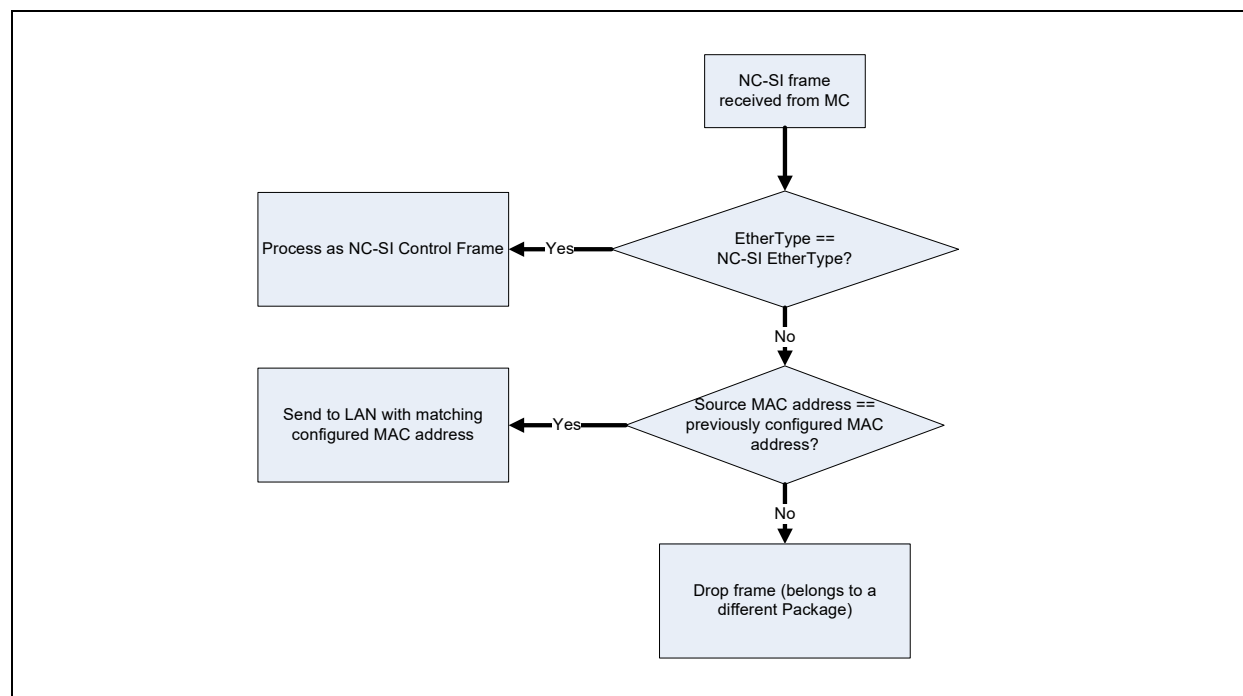


Figure 10-7. NC-SI Frames Receive Flow for the NC

10.6.2 NC-SI Standard Support

10.6.2.1 Supported Features

The E610 supports all the mandatory features of the NC-SI specification. Table 10-11 lists the support for standard NC-SI commands.

The E610 supports NC-SI 1.1.0

Table 10-12 lists optional features supported and the level of support for partially supported commands.

Table 10-11. NC-SI Commands Support

Command	Supported over RMII	Supported over MCTP with Pass Through	Supported over MCTP without Pass Through
Clear initial state	Yes	Yes	Yes
Get Version ID	Yes	Yes	Yes
Get Parameters	Yes	Yes	Yes
Get Controller Packet Statistics	Yes, partially	Yes, partially	Yes, partially
Get Link Status	Yes	Yes	Yes

Table 10-11. NC-SI Commands Support [continued]

Command	Supported over RMI	Supported over MCTP with Pass Through	Supported over MCTP without Pass Through
Enable Channel	Yes	Yes	Yes
Disable Channel	Yes	Yes	Yes
Reset Channel	Yes	Yes	Yes
Enable VLAN	Yes ^{1,2}	Yes ¹	No ³
Disable VLAN	Yes	Yes	No ³
Enable Broadcast Filter	Yes	Yes	No ³
Disable Broadcast Filter	Yes	Yes	No ³
Set MAC Address	Yes	Yes	No ³
Get NC-SI Statistics	Yes	Yes	Yes
Set NC-SI Flow-Control	Yes	No	No ³
Set Link Command	Yes	Yes	Yes
Enable Global multicast Filter	Yes	Yes	No ³
Disable Global multicast Filter	Yes	Yes	No ³
Get Capabilities	Yes	Yes	Yes ⁴
Set VLAN Filters	Yes	Yes	No ³
AEN Enable	Yes	Yes	Yes
Get NC-SI Pass-Through Statistics	Yes, partially	Yes, partially	No ³
Select Package	Yes	Yes	Yes
Deselect Package	Yes	Yes	Yes
Enable Channel Network TX	Yes	Yes	No
Disable Channel Network TX	Yes	Yes	No
Get Package Status ⁵	Yes	No	No
Get Package UUID ⁵	Yes	Yes	Yes
PLDM Request ⁵	No	No	No
OEM Command ⁶	Yes	Yes	Yes
Get Supported Media	No	Yes	Yes
Transport Specific AEN Enable	No	Yes	Yes
Get MC MAC Address ⁷	Yes	Yes	Yes

1. In cases that one of the LAN devices is assigned for the sole use of the manageability and its LAN PCIe function is disabled, using the NC-SI Set Link command while advertising multiple speeds and enabling auto-negotiation, results in the lowest possible speed chosen. To enable link of higher a speed, the MC should not advertise speeds that are below the desired link speed. When doing it, changing the power state of the LAN device has no effect and the link speed is not re-negotiated.
2. The E610 does not support filtering of *User Priority/CFI* bits of VLAN.
3. In MCTP without PT mode, only control commands are supported and not PT traffic. Thus many of the regular NC-SI commands are not supported or are supported in a limited manner, only to enable control and status reporting for the device.
4. When PT is disabled, the Get Capabilities command does not expose all the filtering capabilities of the device.
5. Only relevant in NC-SI 1.1
6. See [Section 10.6.3.2](#) for details.
7. This command is defined in DSP0222 NC-SI 1.2 draft

Table 10-12. Optional NC-SI Features Support

Feature	Implement	Details
AENs	Yes	The following AENs are supported: • Link Change AEN • Configuration Required AEN • Driver Status Change AEN • Medium Change AEN (over MCTP) • Initialization error AEN (Intel OEM AEN) The Driver state AEN might be emitted up to 1 minute after actual driver change if the driver was taken down unexpectedly.
Get Controller Packet Statistics command	Yes, partially ¹	Supports the following counters²: 0-8, 11-16, 21-36³ The statistics are cleared on read, but aside from a reset, are not cleared between reads. Note: The packets counted by counter #35 (1523 - 9022 Byte Frames transmitted) are up to 9522 bytes and not 9022 as requested in the spec. If a core reset occurs between reads of statistics, the "Counters Cleared From Last Read" flags are set in the response.
Get NC-SI statistics	Yes	Support all the counters⁴ Note: If a core reset occurs between reads of statistics, the reported statistics may be wrong.
Get NC-SI Pass-Through Statistics	Yes, partially	Support the following counters: 1, 6, 7.
VLAN Modes	Yes, partially	Support only modes 1, 3.
Buffering Capabilities	Yes	8 Kb
MAC Address Filters	Yes	Supports 2 MAC Addresses per port.
Channel Count	Yes	Supports eight channels.
VLAN Filters	Yes	Support 8 VLAN filters per port. Filtering is ignoring the CFI bit and the 802.1P priority bits
Broadcast Filters	Yes	Support the following filters: • ARP • DHCP • Net BIOS
Multicast Filters	Yes	Supports the following filters: • IPv6 neighbor advertisement • IPv6 router advertisement • DHCPv6 relay and server multicast In NC-SI 1.1 mode, the following filters are also supported • DHCPv6 multicasts from server to clients listening on well-known UDP ports • IPv6 MLD • IPv6 Neighbor Solicitation
Hardware Arbitration	Yes	Supports NC-SI hardware arbitration.

1. Not supported in switch mode

2. *TCTL.EN* should be set to 1b to activate Tx-related counters and *RCTL.RXEN*, *PRT_MNG_MANC.RCV_EN* or *WUC.APME* should be set to enable Rx-related counters.

3. As described in the Get Controller Packet Statistics Counter Numbers table in NC-SI specification.

4. The E610 does not increment the NC-SI control packets dropped counter when packets with checksum errors are dropped. In this case, only the NC-SI command checksum errors counter is updated.

10.6.2.1.1 Error Conditions

In case the device is not able to respond to a command within the 150 ms limit (three 50 ms retries), it will respond with a response code of command unavailable (0x02) and a reason code of package not ready (0x04)

10.6.2.2 AEN Handling

Asynchronous events might occur when the device is not allowed to send them via the AEN Enable command. The following rules defines the behavior of the E610 in these cases:

1. While the device is disabled, for each type of AEN only the last event is kept.
2. Outstanding AENs that occurred while a package was deselected is transmitted when a package is selected.
3. On a transition from channel disabled to channel enabled, all outstanding events are erased to prevent stale event notifications.
4. If the AEN becomes outdated before being sent (for example a link down, link up sequence occurring before the AEN is sent), then no AEN is sent.

Note: AEN is handled independently per PF (e.g. if PF1 is disabled you would still see this AEN when unloading/loading the driver on PF0)

10.6.2.2.1 Driver Status Change AEN Generation

The Host NC Driver Status changes and the Driver Status Change AEN is emitted in the following conditions:

Driver is Active: When a “Driver Version” AQ command is received from the PF associated with this channel.

Driver is Not Active: When CTRL_EXT.DRV_LOAD is set.

10.6.2.3 NC-SI 1.1 New Features

The following features are available when NC-SI version is set to 1.1 in the NC-SI version flag in the NVM NC-SI Configuration 1 word:

- Broadcast and Multicast packets reception is enabled by default at initial state after channel is enabled.
- The Get Package UUID command and Get Package Status command are supported. The UUID reported is the same as described in [Section 10.7.7.1.2](#).
- New format for the Set Link and Get Link commands including support for new speeds.
- The following filters are added to the Enable Global Multicast Filter command:
 - DHCPv6 multicast packets from server to clients listening on well-known UDP ports
 - IPv6 MLD
 - IPv6 Neighbor Solicitation
- New format for the Capabilities Flags word in Get Capabilities response packet.

10.6.3 External Link Control via NC-SI

10.6.3.1 NC-SI Link State Control

In NC-SI mode, the device might dynamically change the PHY power mode according to the NC-SI channel state assuming no other functionality requires the PHY to be active (host or wake up).

The following algorithm is used to define if PHY activity is required:

- At initialization time, the PHY is required to be active only if the *EMP_LINK_ON* bit in Common Firmware Parameters 2 NVM word is set.
- Once a channel is enabled via a Enable Channel NC-SI command, The PHY is powered up.
- If the channel is disabled via a Disable Channel command with the *ALD* bit set, the PHY is disabled.
- If the channel is disabled via a Reset Channel command, the PHY power state is set back to the initial value as define by the *EMP_LINK_ON* bit.

Note: Before transitioning to D3 it is the responsibility of the software device driver to request the PHY to be active for wake-up activities.

10.6.3.2 Set Link Error Codes

The following rules are used to define the error code returned for a Set Link command in case an invalid configuration is requested:

1. Host Driver Check: If a host device driver is present, return a Command fail Response (0x1) with a Set Link Host OS/Driver Conflict Reason (0x0901).
2. Speed Present Check: If no speed is selected, return a General Reason Code for a failed command (0x1) with Parameter Is Invalid, Unsupported, or Out-of-Range Reason (0x2).
3. Parameter Validity:
 - a. Auto-negotiation Parameter Validation: If Auto-negotiation is requested and none of the selected parameters are valid for the device, return a General Reason Code for a failed command (0x1) with a Parameter Is Invalid, Unsupported, or Out-of-Range Reason (0x2).

Note: This means, for example, a command requesting 10 GbE on a 1 GbE device succeeds provided that the command requests at least one other supported speed.

Note: The setting of auto-negotiation enabled field is ignored. For speed and connection types that accept only force mode, force mode is used and for modes that supports only AN, the AN is used to enforce a speed by negotiating only this speed. There are no modes supported by the E610 that supports both AN and force modes.

The same goes for an unsupported duplex setting (a device with no HD support accepts a command with both FD and HD set), and also for HD being requested with speeds of 1 GbE and higher as long as a speed below 1 GbE is also requested (and is supported in HD). The device simply ignores the unsupported parameters.

- b. Force Mode Parameter Validation:

- If more than one link speed is being forced, then return a General Reason Code for a failed command (0x1) and a Command Specific Reason with a Set Link Speed Conflict Error (0x0905).

- If more than one duplex setting is being forced, then return a General Reason Code for a failed command (0x1) with Parameter Is Invalid, Unsupported, or Out-of-Range Reason (0x2).
 - If 1 GbE and above is requested with HD, then return a General Reason Code for a failed command (0x1) and a Command Specific Reason with Set Link Parameter Conflict Reason (0x0903).
4. Media Type Compatibility Check: If current media type is not compatible for the requested link parameters, return a General Reason Code for a failed command (0x1) and a Command Specific Reason with Set Link Media Conflict Error (0x0902).
 5. Power State Compatibility Check: If current power state does not allow for the requested link parameters, return a General Reason Code for a failed command (0x1) and a Command Specific Reason with Set Link Power Mode Conflict Reason (0x0904).
 6. If for some reason the hardware cannot perform the flow required for the command, return a General Reason Code for a failed command (0x1) and a Command Specific Response with Link Command Failed-Hardware Access Error (0x0906).

10.6.3.3 MC External Link Control

The MC can use the NC-SI Set Link command to control the external interface link settings. This command enables the MC to set the auto-negotiation, link speed, duplex, and other parameters.

This command is only available when the host operating system is not present. Indicating the host operating system status can be obtained via the Get Link Status command and/or Host OS Status Change AEN command.

Recommendation:

- Unless explicitly needed, it is not recommended to use this feature. The NC-SI Set Link command does not expose all the possible link settings and/or features. This might cause issues under different scenarios. Even if you decided to use this feature, use it only if the link is down (trust the E610 until proven otherwise).
- It is recommended that the MC first query the link status using the Get Link Status command. The MC should then use this data as a basis and change only the needed parameters when issuing the Set Link command.

For details, refer to the NC-SI specification.

10.6.3.3.1 Set Link While LAN PCIe Functionality is Disabled

In cases where the E610 is used solely for manageability and its LAN PCIe function is disabled, using the NC-SI Set Link command while advertising multiple speeds and enabling auto-negotiation results in the lowest possible speed chosen.

To enable a higher link speed, the MC should not advertise speeds that are below the desired link speed, as the lowest advertised link speed is chosen.

When the E610 is only used for manageability and the link speed advertisement is configured by the MC, changes in the power state of the LAN device is not effected and the link speed is not re-negotiated by the LAN device.

10.6.4 NC-SI Mode — Intel-Specific Commands

In addition to regular NC-SI commands, the following Intel vendor specific commands are supported. The purpose of these commands is to provide a means for the MC to access some of the Intel-specific features present in the E610.

10.6.4.1 Overview

The following features are available via the NC-SI OEM specific commands:

- Receive filters:
- Packet addition decision filters 0x0...0x4
- Packet reduction decision filters 0x5...0x7
- PRT_MNG_MNGONLY register (controls the forwarding of manageability packets to the host)
- Flex 128B filters
- Flex TCP/UDP port filters 0x0...0x2
- IPv4/IPv6 filters
- Get system MAC Address — This command enables the MC to retrieve the system MAC Address used by the MC. This MAC Address can be used for a shared MAC Address mode.
- Keep PHY link up (Veto bit) enable/disable — This feature enables the MC to block PHY reset, which might cause session loss.
- TCO reset — Enables the MC to reset the E610.
- Checksum offloading — Offloads IP/UDP/TCP checksum checking from the MC.
- OS2BMC control commands.
- Firmware version commands.
- Shared MAC and shared IP commands.
- Temperature reporting

These commands are designed to be compliant with their corresponding SMBus commands (if existing). All of the commands are based on a single DMTF defined NC-SI command, known as OEM Command. This command is as follows.

10.6.4.1.1 OEM Command (0x50)

The OEM command can be used by the MC to request the sideband interface to provide vendor-specific information. The Vendor Enterprise Number (VEN) is the unique MIB/SNMP private enterprise number assigned by IANA per organization. Vendors are free to define their own internal data structures in the vendor data fields.

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...	Intel Command Number	Optional Data		
...	...			
...	Optional Data		Padding to 32 bits (0x00)	
...	Checksum			

10.6.4.1.2 OEM Response (0xD0)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	Intel Command Number	Optional Return Data		
...	...			
...	Optional Return Data		Padding to 32 bits (0x00)	
...	Checksum			

Note: Responses have no command-specific reason code, unless otherwise specified within the command.

Note: The commands/responses described as follows includes only the part up to the data. The padding and checksum are implied.

10.6.4.2 OEM Commands Summary

Table 10-13. OEM Specific Command Response Reason Codes

Response Code		Reason Code	
Value	Description	Value	Description
0x1	Command Failed	0x5081	Invalid Intel command number.
		0x5082	Invalid Intel command parameter number.
		0x5085	Internal network controller error.
		0x5086	Invalid vendor enterprise code.
		0x05089	Information Not available
		0x508D	Returned when one of the shared IP commands is received with an out of range resource (IP, port, binding) index.
		0x508E	Returned when a request to disable a port or an IP Address used in a active binding is received.
		0x5090	Returned when a binding of a non enabled resource (MAC, VLAN, IP Address, port) is required.
		0x5091	Returned when the Set Port command is received with an unsupported protocol.
		0x5092	Not in shared mode. Returned when shared mode commands are used while not in shared MAC/IP mode.
		0x008E	Returned when a request to disable a VLAN or a MAC Address used in a active binding is received.

Table 10-14. OEM Commands Summary

Intel Command	Parameter	Command Name	Supported in MCTP without PT	Section
0x00	0x00	Set IP Filters Control	No	10.6.4.3
0x01	0x00	Get IP Filters Control	No	10.6.4.4
0x02	0x0F	Set Manageability Only	No	10.6.4.5.3
	0x10	Set Flexible 128 Filter Mask and Length		10.6.4.5.5
	0x11	Set Flexible 128 Filter Data		10.6.4.5.7
	0x63	Set Flex TCP/UDP Port Filters		10.6.4.5.10
	0x64	Set Flex IPv4 Address Filters		10.6.4.5.12
	0x65	Set Flex IPv6 Address Filters		10.6.4.5.14
	0x67	Set EtherType Filter		10.6.4.5.16
	0x68	Set Packet Addition Extended Filter		10.6.4.5.18
	0x69	Reserved		

Table 10-14. OEM Commands Summary [continued]

Intel Command	Parameter	Command Name	Supported in MCTP without PT	Section
0x03	0x0F	Get Manageability Only	No	10.6.4.6.3
	0x10	Get Flexible 128 Filter Mask and Length		10.6.4.6.5
	0x11	Get Flexible 128 Filter Data		10.6.4.6.7
	0x63	Get Flex TCP/UDP Port Filters		10.6.4.6.10
	0x64	Get Flex IPv4 Address Filters		10.6.4.6.12
	0x65	Get Flex IPv6 Address Filters		10.6.4.6.14
	0x67	Get EtherType Filter		10.6.4.6.16
	0x68	Get Packet Addition Extended Filter		10.6.4.6.18
	0x69	Get Special Filter Modifiers		10.6.4.6.20
0x04	0x10	Set Extended Unicast Packet Reduction	No	10.6.4.7.3
	0x11	Set Extended Multicast Packet Reduction		10.6.4.7.5
	0x12	Set Extended Broadcast Packet Reduction		10.6.4.7.7
0x05	0x10	Get Extended Unicast Packet Reduction	No	10.6.4.8.1
	0x11	Get Extended Multicast Packet Reduction		10.6.4.8.3
	0x12	Get Extended Broadcast Packet Reduction		10.6.4.8.5
0x06	N/A	Get System MAC Address	Yes	10.6.4.9
0x20	N/A	Set Intel Management Control	No	10.6.4.10
0x21	N/A	Get Intel Management Control	No	10.6.4.11
0x22	N/A	TCO Reset	Yes	10.6.4.12
0x23	N/A	Enable IP/UDP/TCP Checksum Offloading	No	10.6.4.13.1
0x24	N/A	Disable IP/UDP/TCP Checksum Offloading	No	10.6.4.13.3
0x25		Reserved		
0x40	0x01	Enable OS2BMC flow	No	10.6.4.15.1
	0x02	Enable Network to BMC flow		10.6.4.15.3
	0x03	Enable Both Network to BMC and Host to BMC flow		10.6.4.15.5
0x40	0x04	Set BMC IP Address	No	10.6.4.15.7
0x41	N/A	Get OS2BMC parameters	No	10.6.4.15.9
0x48	0x1	Get Controller Information	Yes	10.6.4.16.1
0x4B	N/A	Get ASIC Temperature	Yes	10.6.4.17
	0x2	Get SFF Module Temperature	Yes	10.6.4.18

Note: All the commands are supported both over RMII NC-SI and over MCTP.

10.6.4.3 Set Intel Filters Control — IP Filters Control Command (Intel Command 0x00, Filter Control Index 0x00)

This command controls different aspects of the Intel filters.

10.6.4.3.1 Set Intel Filters Control — IP Filters Control Command

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x00	0x00	IP Filters Control (3-2)	
24...25	IP Filters Control (1-0)			

Where IP Filters Control has the following format.

Bit #	Name	Description	Default Value
0	IPv4/IPv6 Mode	IPv6 (0b) = There are zero IPv4 filters and four IPv6 filters. IPv4 (1b) = There are four IPv4 filters and four IPv6 filters.	1b
1...31	Reserved		

Note: This command is kept for compatibility with other projects and has no effect in E610.

10.6.4.3.2 Set Intel Filters Control — IP Filters Control Response

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x00	0x00		

10.6.4.4 Get Intel Filters Control Commands (Intel Command 0x01)

10.6.4.4.1 Get Intel Filters Control — IP Filters Control Command (Intel Command 0x01, Filter Control Index 0x00)

This command controls different aspects of the Intel filters.

Bits				
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x01	0x00		

10.6.4.4.2 Get Intel Filters Control — IP Filters Control Response (Intel Command 0x01, Filter Control Index 0x00)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x01	0x00	IP Filters Control (3-2)	
28...29	IP Filters Control (1-0)			

Note: This command is kept for compatibility with other projects and returns always 0x1 in E610.

10.6.4.5 Set Intel Filters Formats

10.6.4.5.1 Set Intel Filters Command (Intel Command 0x02)

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x02	Parameter Number	Filters Data (Optional)	

10.6.4.5.2 Set Intel Filters Response (Intel Command 0x02)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x02	Filter Control Index	Return Data (Optional)	

10.6.4.5.3 Set Intel Filters — Manageability Only Command (Intel Command 0x02, Filter Parameter 0x0F)

This command sets the PRT_MNG_MNGONLY register. The PRT_MNG_MNGONLY register controls whether PT packets destined to the MC are not forwarded to the Host operating system. The PRT_MNG_MNGONLY register is listed in [Table 10-8](#).

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x02	0x0F	Manageability Only (3-2)	
24...25	Manageability Only (1-0)			

10.6.4.5.4 Set Intel Filters — Manageability Only Response (Intel Command 0x02, Filter Parameter 0x0F)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x0F		

10.6.4.5.5 Set Intel Filters — Flex Filter Enable Mask and Length Command (Intel Command 0x02, Filter Parameter 0x10)

The following command sets the Intel flex filters mask and length.

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x02	0x10	Mask Byte 1	Mask Byte 2
24...27	Mask Byte 3	Mask Byte 4	Mask Byte 5	Mask Byte 6
28...31	Mask Byte 7	Mask Byte 8	Mask Byte 9	Mask Byte 10
32...35	Mask Byte 11	Mask Byte 12	Mask Byte 13	Mask Byte 14
36...39	Mask Byte 15	Mask Byte 16	Reserved	
40	Length			

10.6.4.5.6 Set Intel Filters — Flex Filter Enable Mask and Length Response (Intel Command 0x02, Filter Parameter 0x10)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x10		

10.6.4.5.7 Set Intel Filters — Flex Filter Data Command (Intel Command 0x02, Filter Parameter 0x11)

Table 10-15. Filter Data Group

Code	Bytes Programmed	Filter Data Length
0x0	Bytes 0-29	1 - 30
0x1	Bytes 30-59	1 - 30
0x2	Bytes 60-89	1 - 30
0x3	Bytes 90-119	1 - 30
0x4	Bytes 120-127	1 - 8

Bits				
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x02	0x11	Filter Data Group	Filter Data 1
24...25	...	Filter Data N		

Note: Using this command to configure the filters data must be done after the flex filter mask command is issued and the mask is set.

10.6.4.5.8 Set Intel Filters — Flex Filter Data Response (Intel Command 0x02, Filter Parameter 0x11)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x11		

10.6.4.5.9 Set Intel Filters — Packet Addition Decision Filter Command (Intel Command 0x02, Filter Parameter 0x61)

This command is no longer supported. Use the Set Intel Filters - Packet Addition Extended Decision Filter Command (Intel Command 0x02, Filter Parameter 0x68 - [Section 10.6.4.5.18](#)) instead.

10.6.4.5.10 Set Intel Filters — Flex TCP/UDP Port Filter Command (Intel Command 0x02, Filter Parameter 0x63)

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x02	0x63	Port Filter index	TCP/UDP Port MSB
24...25	TCP/UDP Port LSB	Reserved		

Filter index range: 0x0...0xA. When NC-SI 1.1 is supported the range is 0x0...0x9.

If the filter index is larger than 10, a command failed response code is returned with Invalid Intel Parameter Number reason (0x5082).

10.6.4.5.11 Set Intel Filters — Flex TCP/UDP Port Filter Response (Intel Command 0x02, Filter Parameter 0x63)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x63		

10.6.4.5.12 Set Intel Filters — IPv4 Filter Command (Intel Command 0x02, Filter Parameter 0x64)

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x02	0x64	IP Filter Index	IPv4 Address (3)
24...26	IPv4 Address (2-0)			

Filter index range: 0x0...0x3.

10.6.4.5.13 Set Intel Filters — IPv4 Filter Response (Intel Command 0x02, Filter Parameter 0x64)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x64		

If the IP filter index is larger than 3, a command failed response code is returned Invalid Intel Parameter Number reason (0x5082).

10.6.4.5.14 Set Intel Filters — IPv6 Filter Command (Intel Command 0x02, Filter Parameter 0x65)

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x02	0x65	IP Filter Index	IPv6 Address Byte 15 (MSB)
24...27	IPv6 Address Byte 14	IPv6 Address Byte 13	IPv6 Address Byte 12	IPv6 Address Byte 11
28...31	IPv6 Address Byte 10	IPv6 Address Byte 9	IPv6 Address Byte 8	IPv6 Address Byte 7
32...35	IPv6 Address Byte 6	IPv6 Address Byte 5	IPv6 Address Byte 4	IPv6 Address Byte 3
36...38	IPv6 Address Byte 2	IPv6 Address Byte 1	IPv6 Address Byte 0 (LSB)	

10.6.4.5.15 Set Intel Filters — IPv6 Filter Response (Intel Command 0x02, Filter Parameter 0x65)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x65		

If the IP filter index is larger the 3, a command failed Response Code is returned, Invalid Intel Parameter Number reason (0x5082).

10.6.4.5.16 Set Intel Filters - EtherType Filter Command (Intel Command 0x02, Filter Parameter 0x67)

Bits				
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x02	0x67	EtherType Filter Index	EtherType Filter MSB
24...26	...	...	EtherType Filter LSB	

Where the EtherType filter has the format as described in [Section 8.2.2.18.4](#).

10.6.4.5.17 Set Intel Filters - EtherType Filter Response (Intel Command 0x02, Filter Parameter 0x67)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x67		

If the EtherType filter index is different than 2 or 3, a command failed Response Code is returned Invalid Intel Parameter Number reason (0x5082).

10.6.4.5.18 Set Intel Filters - Packet Addition Extended Decision Filter Command (Intel Command 0x02, Filter Parameter 0x68)

See [Figure 10-3](#) for description of the decision filters structure.

The command must overwrite any previously stored value. The value set is not checked.

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x02	0x68	Extended Decision Filter Index	Extended Decision Filter 1 MSB
24...27	...	...	Extended Decision filter 1 LSB	Extended Decision filter 0 MSB
28...30	...	...	Extended Decision filter 0 LSB	

Extended decision filter index range: 0...4

Filter 0: See [Table 10-16](#).

Filter 1: See [Table 10-17](#).

Table 10-16. Filter Values

Bit #	Name	Description
1:0	Unicast (AND)	If set, packets must match unicast filter 0 to 1, respectively.
3:2	Reserved	Reserved
4	Broadcast (AND)	If set, packets must match the broadcast filter.
12:5	VLAN (AND)	If set, packets must match VLAN filter 0 to 7, respectively.
16:13	IPv4 Address (AND)	If set, packets must match IPv4 filter 0 to 3, respectively
20:17	IPv6 Address (AND)	If set, packets must match IPv4 filter 0 to 3, respectively
22:21	Unicast (OR)	If set, packets can pass if match unicast filter 0 to 3, respectively or a different OR filter.
24:23	Reserved	Reserved
25	Broadcast (OR)	If set, packets can pass if match the broadcast filter or a different OR filter.
26	Multicast (AND)	If set, packets must match the multicast filter.
27	ARP Request (OR)	If set, packets can pass if match the ARP request filter or a different OR filter.
28	ARP Response (OR)	If set, packets can pass if match the ARP response filter or a different OR filter.
29	Neighbor Discovery - 134 (OR)	If set, packets can pass if match the neighbor discovery filter(type134 - router advertisement) or a different OR filter.
30	Port 0x298 (OR)	If set, packets can pass if match a fixed TCP/UDP port 0x298 filter or a different OR filter.
31	Port 0x26F (OR)	If set, packets can pass if match a fixed TCP/UDP port 0x26F filter or a different OR filter.

Table 10-17. Extended Filter 1 Values

Bit #	Name	Description
3:0	EtherType 0 -3 (AND)	If set, packets must match the EtherType filter 0 to 3, respectively.
7:4	EtherType 0 -3 (OR)	If set, packets must match the EtherType filter 0 to 3, respectively or a different OR filter.
18:8	Flex port 10:0 (OR)	If set, packets can pass if match the TCP/UDP Port filter 10:0.
19	DHCPv6 (OR)	If set, packets can pass if match the DHCPv6 port (0x0223).
20	DHCP Client (OR)	If set, packets can pass if match the DHCP Server port (0x0043).
21	DHCP Server (OR)	If set, packets can pass if match the DHCP Client port (0x0044).
22	NetBIOS Name Service (OR)	If set, packets can pass if match the NetBIOS Name Service port (0x0089).
23	NetBIOS Datagram Service (OR)	If set, packets can pass if match the NetBIOS Datagram Service port (0x008A).
24	Flex TCO (OR)	If set, packets can pass if match the Flex 128 TCO filter.
25	Neighbor Discovery - 135 (OR)	If set, packets must also match the neighbor discovery filter (type135 - Neighbor Solicitation). or a different OR filter.
26	Neighbor Discovery - 136 (OR)	If set, packets must also match the neighbor discovery filter (type136 - Neighbor Advertisement) or a different OR filter.
27	Neighbor Discovery - 137 (OR)	If set, packets must also match the neighbor discovery filter (type137 - Redirect) or a different OR filter.
28	Reserved	Controls the inclusion of ICMPv4 filtering in the manageability filter decision (OR section).
29	MLD	If set, packets must also match one of the MLD ICMPv6 types or a different OR filter.
31:30	Reserved	Reserved

10.6.4.5.19 Set Intel Filters – Packet Addition Extended Decision Filter Response (Intel Command 0x02, Filter Parameter 0x68)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x68		

If the extended decision filter index is larger than 5, a command failed Response Code is returned Invalid Intel Parameter Number reason (0x5082).

10.6.4.6 Get Intel Filters Formats

10.6.4.6.1 Get Intel Filters Command (Intel Command 0x03)

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x03	Parameter Number		

10.6.4.6.2 Get Intel Filters Response (Intel Command 0x03)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	Parameter Number	Options Return Data	

10.6.4.6.3 Get Intel Filters — Manageability Only Command (Intel Command 0x03, Filter Parameter 0x0F)

This command retrieves the PRT_MNG_MNGONLY register. The PRT_MNG_MNGONLY register controls whether PT packets destined to the MC are also be forwarded to the host operating system.

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x03	0x0F		

10.6.4.6.4 Get Intel Filters — Manageability Only Response (Intel Command 0x03, Filter Parameter 0x0F)

The PRT_MNG_MNGONLY register structure is listed in [Table 10-8](#).

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x0F	Manageability to Host (3-2)	
28...29	Manageability to Host (1-0)			

10.6.4.6.5 Get Intel Filters — Flex Filter 0 Enable Mask and Length Command (Intel Command 0x03, Filter Parameter 0x10)

The following command retrieves the Intel flex filters mask and length. See [Section 10.4.3.5](#) for details of the values returned by this command.

Bits				
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x03	0x10		

10.6.4.6.6 Get Intel Filters — Flex Filter 0 Enable Mask and Length Response (Intel Command 0x03, Filter Parameter 0x10)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x10	Mask Byte 1	Mask Byte 2
28...31	Mask Byte 3	Mask Byte 4	Mask Byte 5	Mask Byte 6
32...35	Mask Byte 7	Mask Byte 8	Mask Byte 9	Mask Byte 10
36...39	Mask Byte 11	Mask Byte 12	Mask Byte 13	Mask Byte 14
40...43	Mask Byte 15	Mask Byte 16	Reserved	
44	Flexible Filter Length			

10.6.4.6.7 Get Intel Filters — Flex Filter 0 Data Command (Intel Command 0x03, Filter Parameter 0x11)

The following command retrieves the Intel flex filters data.

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...22	0x03	0x11	Filter Data Group 0...4	

The filter data group parameter defines which bytes of the flex filter are returned by this command:

Table 10-18. Filter Data Group

Code	Bytes Returned
0x0	Bytes 0-29
0x1	Bytes 30-59
0x2	Bytes 60-89
0x3	Bytes 90-119
0x4	Bytes 120-127

10.6.4.6.8 Get Intel Filters — Flex Filter 0 Data Response (Intel Command 0x03, Filter Parameter 0x11)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x11	Filter Group Number	Filter Data 1
28...29	...	Filter Data N		

10.6.4.6.9 Get Intel Filters — Packet Addition Decision Filter Command (Intel Command 0x03, Filter Parameter 0x61)

This command is no longer supported. Use the Get Intel Filters - Packet Addition Extended Decision Filter Command (Intel Command 0x03, Filter Parameter 0x68 - [Section 10.6.4.6.18](#)) instead.

10.6.4.6.10 Get Intel Filters — Flex TCP/UDP Port Filter Command (Intel Command 0x03, Filter Parameter 0x63)

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...22	0x03	0x63	TCP/UDP Filter Index	

Filter index range: 0x0...0x2.

10.6.4.6.11 Get Intel Filters — Flex TCP/UDP Port Filter Response (Intel Command 0x03, Filter Parameter 0x63)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x63	TCP/UDP Filter Index	TCP/UDP Port (1)
28...29	TCP/UDP Port (0)	Port Flags		

Filter index range: 0x0...0x2.

10.6.4.6.12 Get Intel Filters — IPv4 Filter Command (Intel Command 0x03, Filter Parameter 0x64)

Bits				
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...22	0x03	0x64	IPv4 Filter Index	

Note: The filters index range can vary according to the IPv4/IPv6 mode setting in the Filters Control command.

IPv4 Mode: Filter index range: 0x0...0x3.

IPv6 Mode: This command should not be used in IPv6 mode.

10.6.4.6.13 Get Intel Filters — IPv4 Filter Response (Intel Command 0x03, Filter Parameter 0x64)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x64	IPv4 Filter Index	IPv4 Address (3)
28...30	IPv4 Address (2-0)			

10.6.4.6.14 Get Intel Filters — IPv6 Filter Command (Intel Command 0x03, Filter Parameter 0x65)

Bits				
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...22	0x03	0x65	IPv6 Filter Index	

Note: The filters index range can vary according to the IPv4/IPv6 mode setting in the Filters Control command

IPv4 Mode: Filter index range: 0x0...0x2.

IPv6 Mode: Filter index range: 0x0...0x3.

10.6.4.6.15 Get Intel Filters – IPv6 Filter Response (Intel Command 0x03, Filter Parameter 0x65)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x65	IPv6 Filter Index	IPv6 Address Byte 15 (MSB)
28...31	IPv6 Address Byte 14	IPv6 Address Byte 13	IPv6 Address Byte 12	IPv6 Address Byte 11
32...35	IPv6 Address Byte 10	IPv6 Address Byte 9	IPv6 Address Byte 8	IPv6 Address Byte 7
36...39	IPv6 Address Byte 6	IPv6 Address Byte 5	IPv6 Address Byte 4	IPv6 Address Byte 3
40...42	IPv6 Address Byte 2	IPv6 Address Byte 1	IPv6 Address Byte 0 (LSB)	

10.6.4.6.16 Get Intel Filters - EtherType Filter Command (Intel Command 0x03, Filter Parameter 0x67)

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...22	0x03	0x67	EtherType Filter Index	

Valid indices: 0...3

10.6.4.6.17 Get Intel Filters - EtherType Filter Response (Intel Command 0x03, Filter Parameter 0x67)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x67	EtherType Filter Index	EtherType Filter MSB
28...30	...	...	EtherType Filter LSB	

If the EtherType filter index is larger than 3, a command failed Response Code is returned Invalid Intel Parameter Number reason (0x5082).

10.6.4.6.18 Get Intel Filters – Packet Addition Extended Decision Filter Command (Intel Command 0x03, Filter Parameter 0x68)

This command enables the MC to retrieve the extended decision filter.

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...22	0x03	0x68	Extended Decision Filter Index	

10.6.4.6.19 Get Intel Filters – Packet Addition Extended Decision Filter Response (Intel Command 0x03, Filter Parameter 0x68)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x68	Decision Filter Index	Decision Filter 1 MSB
28...31	...	...	Decision Filter 1 LSB	Decision Filter 0 MSB
32...34	...	...	Decision Filter 0 LSB	

Where decision filter 0 and decision filter 1 have the structure as detailed in the respective Set commands.

If the extended decision filter index is larger than 4, a command failed Response Code is returned Invalid Intel Parameter Number reason (0x5082).

10.6.4.6.20 Get Intel Filters – Special Modifier Command (Intel Command 0x03, Filter Parameter 0x69)

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x03	0x69	Padding	

10.6.4.6.21 Get Intel Filters - Special Modifier Response (Intel Command 0x02, Filter Parameter 0x69)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x69	Special Modifier Register MSB	
28...31	Special Modifier Register LSB		Padding	

10.6.4.7 Set Intel Packet Reduction Filters Formats

The non-extended commands are obsolete. The extended commands ([Section 10.6.4.7.3](#) to [Section 10.6.4.7.8](#)) should be used instead.

10.6.4.7.1 Set Intel Packet Reduction Filters Command (Intel Command 0x04)

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x04	Packet Reduction Index	Packet Reduction Data	

Note: It is advised that the MC only use the extended packet reduction commands.

The *Packet Reduction* data field has the following structure:

Table 10-19. Packet Reduction Field Description

Bit #	Name	Description
12:0	Reserved	Reserved
16:13	IPv4 Address (AND)	If set, packets must match IPv4 filter 0 to 3, respectively.
20:17	IPv6 Address (AND)	If set, packets must match IPv4 filter 0 to 3, respectively.
27:21	Reserved	Reserved.

Table 10-19. Packet Reduction Field Description [continued]

Bit #	Name	Description
28	ARP Response (OR)	If set, packets can pass if match the ARP response filter or a different OR filter.
29	Reserved	Reserved.
30	Port 0x298	If set, packets can pass if match a fixed TCP/UDP port 0x298 filter.
31	Port 0x26F	If set, packets can pass if match a fixed TCP/UDP port 0x26F filter.

Table 10-20. Extended Packet Reduction Field Description

Bit #	Name	Description
3:0	EtherType 0 -3 (AND)	If set, packets must match the EtherType filter 0 to 3, respectively.
7:4	EtherType 0-3 (OR)	If set, packets can pass if match the EtherType filter 0 to 3, respectively.
15:12	Reserved	Reserved.
8:18	Flex port 10:0 (OR)	If set, packets can pass if match the TCP/UDP Port filter 10:0.
23:19	Reserved	Reserved.
24	Flex TCO (OR)	If set, packets can pass if match the Flex 128 TCO filter.
27:25	Reserved	Reserved.
28	ICMPv4	Is set, ICMPv4 packets can pass.
31:29	Reserved	Reserved.

The filtering is divided into two decisions:

- Bit 20:13 in [Table 10-19](#) and Bits 3:2 in [Table 10-20](#) works in an AND manner; it must be true in order for a packet to pass (if was set).

Bits 28 in [Table 10-19](#) and Bits 24:10 in [Table 10-20](#) work in an OR manner; at least one of them must be true for a packet to pass (if any were set).

10.6.4.7.2 Set Intel Packet Reduction Filters Response (Intel Command 0x04)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x04	Packet Reduction Index		

10.6.4.7.3 Set Unicast Extended Packet Reduction Command (Intel Command 0x04, Reduction Filter Index 0x10)

The command has the following format:

Bits				
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x04	0x10	Extended Unicast Reduction Filter MSB	...
24...27	...	Extended Unicast Reduction Filter LSB	Unicast Reduction Filter MSB	...
28...29	...	Unicast Reduction Filter LSB		

The command overwrites any previously stored value.

Note: See [Table 10-19](#) and [Table 10-20](#) for description of the unicast extended packet reduction format.

10.6.4.7.4 Set Unicast Extended Packet Reduction Response (Intel Command 0x04, Reduction Filter Index 0x10)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x04	0x10		

10.6.4.7.5 Set Multicast Extended Packet Reduction Command (Intel Command 0x04, Reduction Filter Index 0x11)

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x04	0x11	Extended Multicast Reduction Filter MSB	...
24...27	...	Extended Multicast Reduction Filter LSB	Multicast Reduction Filter MSB	...
28...29	...	Multicast Reduction Filter LSB		

Note: See [Table 10-19](#) and [Table 10-20](#) for description of the multicast extended packet reduction format.

The command overwrites any previously stored value.

10.6.4.7.6 Set Multicast Extended Packet Reduction Response (Intel Command 0x04, Reduction Filter Index 0x11)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x04	0x11		

10.6.4.7.7 Set Broadcast Extended Packet Reduction Command (Intel Command 0x04, Reduction Filter Index 0x12)

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x04	0x12	Extended Broadcast Reduction Filter MSB	...
24...27	...	Extended Broadcast Reduction Filter LSB	Broadcast Reduction Filter MSB	...
28...29	...	Broadcast Reduction Filter LSB		

Note: See [Table 10-19](#) and [Table 10-20](#) for description of the broadcast extended packet reduction format.

The command overwrites any previously stored value.

10.6.4.7.8 Set Broadcast Extended Packet Reduction Response (Intel Command 0x04, Reduction Filter Index 0x12)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x04	0x12		

10.6.4.8 Get Intel Packet Reduction Filters Formats

The non extended commands are obsolete. Use the extended commands ([Section 10.6.4.8.1](#) to [Section 10.6.4.8.6](#)) instead.

10.6.4.8.1 Get Unicast Extended Packet Reduction Command (Intel Command 0x05, Reduction Filter Index 0x10)

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x05	0x10		

10.6.4.8.2 Get Unicast Extended Packet Reduction Response (Intel Command 0x05, Reduction Filter Index 0x10)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x05	0x10	Extended Unicast Packet Reduction (3-2)	
28...31	Extended Unicast Packet Reduction (1-0)		Unicast Packet Reduction (3-2)	
32...33	Unicast Packet Reduction (1-0)			

10.6.4.8.3 Get Multicast Extended Packet Reduction Command (Intel Command 0x05, Reduction Filter Index 0x11)

Bits				
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x05	0x11		

10.6.4.8.4 Get Multicast Extended Packet Reduction Response (Intel Command 0x05, Reduction Filter Index 0x11)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x05	0x11	Extended Multicast Packet Reduction (3-2)	
28...31	Extended Multicast Packet Reduction (1-0)		Multicast Packet Reduction (3-2)	
32...33	Multicast Packet Reduction (1-0)			

10.6.4.8.5 Get Broadcast Extended Packet Reduction Command (Intel Command 0x05, Reduction Filter Index 0x12)

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x05	0x12		

10.6.4.8.6 Get Broadcast Extended Packet Reduction Response (Intel Command 0x05, Reduction Filter Index 0x12)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x05	0x12	Extended Broadcast Packet Reduction (3-2)	
28...31	Extended Broadcast Packet Reduction (1-0)		Broadcast Packet Reduction (3-2)	
32...33	Broadcast Packet Reduction (1-0)			

10.6.4.9 System MAC Address

10.6.4.9.1 Get System MAC Address Command (Intel Command 0x06)

In order to support a system configuration that requires the NC to hold the MAC Address for the MC (such as shared MAC Address mode), the following command is provided to enable the MC to query the NC for a valid MAC Address.

The NC must return the system MAC Addresses. The MC should use the returned MAC Addressing as a shared MAC Address by setting it using the Set MAC Address command as defined in NC-SI 1.0.

When a single function is defined on the port, it returns the LAN MAC Address of this function as read from the PF allocations NVM section or from the alternate RAM or as set by the Manage MAC Address write AQ command. When more than one function is defined on the port, it returns the address of the lowest defined function on this port.

It is also recommended that the MC use packet reduction and the Manageability-to-Host command to set the proper filtering method.

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20	0x06			

10.6.4.9.2 Get System MAC Address Response (Intel Command 0x06)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x06	MAC Address		
28...30	MAC Address			

10.6.4.10 Set Intel Management Control Formats

10.6.4.10.1 Set Intel Management Control Command (Intel Command 0x20)

Bits				
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...22	0x20	0x00	Intel Management Control 1	

Where Intel Management Control 1 is as follows:

Bit #	Default Value	Description
0	0b	Enable Critical Session Mode (Keep PHY Link Up and Veto Bit). 0b = Disabled 1b = Enabled When critical session mode is enabled, the following behaviors are disabled: - The PHY is not reset on PE_RST# and PCIe resets (in-band and link drop). Other reset events are not affected — Internal_Power_On_Reset, device disable, Force TCO, and PHY reset by software. In any case, CORER events are not blocked. - The PHY does not change its power state. As a result, link speed does not change. - The device does not initiate configuration of the PHY to avoid losing link.

Caution: The System Firmware Update Control option in the command should be supported only over MCTP over SMBus.

Note: The Intel Management Control 1 is also reflected in the Get Intel Management Control Response. In this case, the value of “No change” has no meaning, and the current setting should be reflected.

Note: If the parameter that the BMC sets is not supported by the device, the device returns Response Code 0x1 (Command Failed) with Reason Code 'Invalid Intel command parameter number' (0x5082). The setting of the System Firmware Update Control field should be kept across EMP reset, so it should be kept in the Alternate RAM.

Note: The policy set by this command is applicable for any firmware update channel – either from host or via PLDM firmware update.

10.6.4.10.2 Set Intel Management Control Response (Intel Command 0x20)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x20	0x00		

10.6.4.11 Get Intel Management Control Formats

10.6.4.11.1 Get Intel Management Control Command (Intel Command 0x21)

Bits				
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x21	0x00		

Where Intel Management Control 1 is as described in [Section 10.6.4.10.2](#).

10.6.4.11.2 Get Intel Management Control Response (Intel Command 0x21)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x21	0x00	Intel Management Control 1	

10.6.4.12 TCO Reset

Depending on the bit set in the TCO mode field this command causes the E610 to perform either:

1. TCO reset

- If force TCO reset is enabled in the NVM (see [Section 6.3.49.2](#)). The force TCO reset clears the data path (Rx/Tx) of the E610 to enable the MC to transmit/receive packets through the E610.
- If the MC has detected that the operating system is hung and has blocked the Rx/Tx path, the force TCO reset clears the data-path (Rx/Tx) of the NC to enable the MC to transmit/receive packets through the NC.
- After successfully performing the command, the NC considers the Force TCO command as an indication that the operating system is hung and clears the *DRV_LOAD* flag (disable the driver). If TCO reset is disabled in the NVM, the E610 clears the *CTRL_EXT.DRV_LOAD* bit but does not reset the data path and notifies the MC on successful completion.

2. TCO isolate

- If TCO isolate is enabled in the NVM (see [Section 6.3.49.3](#)). The TCO Isolate command disables PCIe write operations to the LAN port.
- If TCO isolate is disabled in NVM, the E610 does not execute the command but sends a response to the MC with successful completion.
- If TCO isolate is disabled in NVM, the E610 does not execute the command but sends a response to the MC with successful completion.
- Once TCO isolate is set, the driver is supposed to be disabled, and it is reported as such to the MC.

Note: Once set, isolation is active until a TCO reset disables it.

3. Firmware reset

- This command causes re-initialization of all the manageability functions and re-loads of manageability related NVM words (such as firmware patch code).
- When the MC loads a new management related NVM image (like a firmware patch) the Firmware Reset command loads the management related NVM information without the need to power down the system.
- This command is issued to the package and affects all channels. After the firmware reset, the firmware Semaphore register (FWSM) is re-initialized.
- Applying this command resets the entire device and also has an effect on TCO reset.

Note: TCO isolate affects only the channel (port) that the command was issued to. Force TCO resets the entire device (all channels in the package).

Following firmware reset, the MC needs to re-initialize all ports.

Note: Only one of the fields should be set in a given command. Setting more than one field might yield unexpected results.

10.6.4.12.1 Perform Intel TCO Reset Command (Intel Command 0x22)

Bits				
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x22	TCO Mode		

Where TCO mode is:

Field	Bit(s)	Description
DO_TCO_RST	0	Do TCO Reset. 0b = Do nothing. 1b = Perform TCO reset.
DO_TCO_ISOLATE ¹	1	Do TCO Isolate. 0b = Enable PCIe write access to LAN port. 1b = Isolate Host PCIe write operation to the port. Notes: - Should be used for debug only. - The TCO Isolate do not impact MCTP traffic - When isolate is set, the OS2BMC flow is also disabled.
RESET_MGMT	2	Reset Manageability; Re-load Manageability NVM Words. 0b = Do nothing. 1b = Issue firmware reset to manageability. Setting this bit generates a one-time firmware reset. Following the reset, management related data from the NVM is loaded. Note: A reset of the internal firmware causes a reset of the entire device.
Reserved	7:3	Reserved (set to 0x00).

Note: For compatibility, the TCO Reset command without the TCO mode parameter is accepted (TCO reset is done).

- TCO isolate host write operation enabled in the NVM.

10.6.4.12.2 Perform Intel TCO Reset Response (Intel Command 0x22)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24	0x22			

10.6.4.13 Checksum Offloading

This command enables the checksum offloading filters in the NC.

When enabled, these filters block any packets that did not pass IP, UDP or TCP checksum from being forwarded to the MC.

10.6.4.13.1 Enable Checksum Offloading Command (Intel Command 0x23)

Bits				
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20	0x23			

10.6.4.13.2 Enable Checksum Offloading Response (Intel Command 0x23)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
23	0x23			

10.6.4.13.3 Disable Checksum Offloading Command (Intel Command 0x24)

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20	0x24			

10.6.4.13.4 Disable Checksum Offloading Response (Intel Command 0x24)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24	0x24			

10.6.4.14 Shared MAC and Shared IP Support Commands (Intel Command 0x25)

To meet the requirements introduced by sharing IP Addresses, modifications and additions to the NC-SI command set are required. These changes include the new commands in this section and the modifications described in [Section 10.4.6](#).

Note: All indexes in this command set starts at one to match the NC-SI methodology.

10.6.4.15 OS2BMC Configuration

These commands control enabling of the OS2BMC flow.

10.6.4.15.1 Enable OS2BMC Flow Command (Intel Command 0x40, Index 0x1)

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x40	0x01		

10.6.4.15.2 Enable OS2BMC Flow Response (Intel Command 0x40, Index 0x1)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x40	0x01		

10.6.4.15.3 Enable Network-to-BMC Flow Command (Intel Command 0x40, Index 0x2)

Bits				
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x40	0x02		

10.6.4.15.4 Enable Network-to-BMC Flow Response (Intel Command 0x40, Index 0x2)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x40	0x02		

10.6.4.15.5 Enable Both Host and Network-to-BMC Flows Command (Intel Command 0x40, Index 0x3)

Bits				
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x40	0x03		

10.6.4.15.6 Enable Both Host and Network-to-BMC Flows Response (Intel Command 0x40, Index 0x3)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x40	0x03		

10.6.4.15.7 Set MC IP Address Command (Intel Command 0x40, Index 0x4)

This command is supported by the E610 for legacy support purposes, but no action is taken upon reception. The IP Address is not stored and when a Get OS2BMC Parameters command is received, the IP valid flag is cleared.

The IP type entry indicate whether the IP Address is an IPv4 or an IPv6 address:

0 = IPv4

1 = IPv6

2 = No IP Address, then the command should not include an IP Address.

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x40	0x04	IP Type	IPv6 Address (Byte 15 - MSB)/IPv4 Address (Byte 3 - MSB)
24...27	IPv6 Address (Byte 14)/ IPv4 Address (Byte 2)	IPv6 Address (Byte 13)/ IPv4 Address (Byte 1)	IPv6 Address (Byte 12)/ IPv4 Address (Byte 0 - LSB)	IPv6 Address (Byte 11)/ Reserved
28...31	IPv6 Address (Byte 10)/ Reserved	IPv6 Address (Byte 9)/ Reserved	IPv6 Address (Byte 8)/ Reserved	IPv6 Address (Byte 7)/ Reserved
32...35	IPv6 Address (Byte 6)/ Reserved	IPv6 Address (Byte 5)/ Reserved	IPv6 Address (Byte 4)/ Reserved	IPv6 Address (Byte 3)/ Reserved
36...38	IPv6 Address (Byte 2)/ Reserved	IPv6 Address (Byte 1)/ Reserved	IPv6 Address (Byte 0 - LSB)/Reserved	

10.6.4.15.8 Set BMC IP Address Response (Intel Command 0x40, Index 0x4)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x40	0x04		

10.6.4.15.9 Get OS2BMC Parameters Command (Intel Command 0x41)

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20	0x41			

10.6.4.15.10 Get OS2BMC Parameters Response (Intel Command 0x41)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x41	Status	IP TypeIPv6 Address (Byte 15 - MSB)/IPv4 Address (Byte 3 - MSB)	IPv6 Address (Byte 14)/IPv4 Address (Byte 2)
28...31	IPv6 Address (Byte 13)/IPv4 Address (Byte 1)	IPv6 Address (Byte 12)/IPv4 Address (Byte 0 - LSB)	IPv6 Address (Byte 11)/Reserved	IPv6 Address (Byte 10)/Reserved
32...35	IPv6 Address (Byte 9)/Reserved	IPv6 Address (Byte 8)/Reserved	IPv6 Address (Byte 7)/Reserved	IPv6 Address (Byte 6)/Reserved
36...39	IPv6 Address (Byte 5)/Reserved	IPv6 Address (Byte 4)/Reserved	IPv6 Address (Byte 3)/Reserved	IPv6 Address (Byte 2)/Reserved
40...41	IPv6 Address (Byte 1)/Reserved	IPv6 Address (Byte 0 - LSB)/Reserved		

Where the status byte partition is as follows:

Table 10-21. Status Byte Description

Bits	Content
0	Relevant only if the IP Address valid bit is set. 0b = IPv4 1b = IPv6
1	IP Address valid. Never valid for the E610.
1:0	Reserved.
2	Network to BMC Status. 0b = Network-to-BMC flow is disabled. 1b = Network-to-BMC flow is enabled.
3	OS2BMC Status. 0b = OS2BMC flow is disabled. 1b = OS2BMC flow is enabled.
7:4	Reserved.

10.6.4.16 Diagnostic Command (Intel Command 0x48)

10.6.4.16.1 Get Controller Information Command (Intel Command 0x48, Index 0x1)

This command gathers the controller identification information and return it back to the MC.

Bytes	Bits			
	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x48	0x01		

10.6.4.16.2 Get Controller Information Response (Intel Command 0x48, Index 0x1)

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x48	0x01	Reserved	Number of Inventory Entries
28...31	Controller Info Item 1 ID	Controller Info Item 1 Length	Controller Info Item 1 Data	
...	...			
...	Controller Info Item 2 ID	Controller Info Item 2 Length	Controller Info Item 2 Data	
...	...			
...	Controller Info Item n ID	Controller Info Item n Length	Controller Info Item n Data	
...	...			

The possible inventory items are described as follows. Note that not all the inventory items would be present in all the implementations of this command.

Table 10-22. Controller Information Items

ID	Length (in Bytes)	Data	Notes
0x00	3	Device ID (2 bytes) + RevID	This is the hardware default value (no value programmed via the NVM).
0x0B	2	NVM Image Version	
0x0C	4	EMP ROM Internal Version	
0x0D	4	EMP Flash Internal version	Same version as in Get Version Admin command.
0x0E	2	PXE firmware version	MajorVersion.MinorVersion.Build.
0x10	2	uEFI firmware version	

10.6.4.17 Get ASIC Temperature Value (Intel Command 0x4B)

This is a pre spec implementation of NC-SI 1.2 Get ASIC Temperature command

Table 10-23. Get ASIC Temperature Command Packet

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x4B	Padding		
24...27	Checksum (3-2)		Checksum (1-0)	
28...31	Padding			
32...35				
36...39				
40...43				
44...45	Padding			

Table 10-24. Get ASIC Temperature Response Packet

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x4B	Reserved	Maximum Temperature	Current Temperature
28...31	Checksum (3-2)		Checksum (1-0)	
32...35	Padding			
36...39				
40...43				
44...45	Padding			

Maximum Temperature Value - This value is the maximum T-Diode temperature limit in degrees Celsius at which the E610 can operate at full load for its rated service lifetime. The value should be de-rated to take measurement tolerance into account. The value shall be reported as a hexadecimal integer number. The value to report for the E610 is: 102 – 0x66 (105 – Thermal sensor accuracy of +/- 3).

Current Temperature Value - This value is the current real-time temperature of the chip in degrees Celsius. The value shall be reported as a hexadecimal integer number.

10.6.4.18 Get SFF Module Temperature Command (Intel Command 0x4B, Parameter 0x02)

The Get SFF Module Temperature Sensor Command allows the Management controller to query for the real-time temperature value and thresholds of the optical transceiver attached to the channel. Implementations that do not support either fixed optics or an SFF-like cage that supports pluggable transceivers that can provide temperature information, such as a Base-T Ethernet adapter, should not implement this command.

Table 10-25. Get SFF Module Temperature Command Packet

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x4B	0x2	Padding	
24...27	Checksum (3-2)		Checksum (1-0)	
28...31	Padding			
32...35				
36...39				
40...43				
44...45	Padding			

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Get SFF Module Temperature command and send a response.

The Get SFF Module Temperature Response frame contains the current temperature of the attached module and the high side temperature thresholds.

Definitions and interpretation of the data fields in the response are defined in the relevant SFF or MSA specification (e.g.: SFF-8472, SFF-8436, SFF-8636 etc.). 16 bit values are encoded as one contiguous entity with the most significant bit in bit 15 (or 31) and least significant bit in bit 0 (or 16) in the response packet.

The command should complete successfully and Intel OEM reason code - *Information not available* - shall be used if the transceiver is not present, does not provide temperature data or if the command is issued before the transceiver has not yet achieved power up state (ref: SFF-8472 address 0xA2 byte 110 bit 0; SFF-8436 address lower 0xA0 byte 2 bit 0; SFF-8636 page 0x0 byte 2 bit 0).

Devices supporting SFF-8472 diagnostics are identified by a non-zero value at address 94 on I²C address 0xA0. For the devices supporting SFF-8636/8436, it is assumed that on customer transceivers, the temperature data is always available.

Table 10-26. Get SFF Module Temperature Response

	Bits			
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI Header			
4...7				
8...11				
12...15				
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x4B	0x02	Reserved	
28...31	Temp High Alarm Threshold		Temp High Warning Threshold	
32...35	Temperature Value		Padding	

Temperature/Threshold fields:

For SFP+ modules - SFF-8472 specification references:

- Temp High Alarm Threshold - the contents of address 0xA2 offset 0-1
- Temp High Warning Threshold - the contents of address 0xA2 offset 4-5
- Temperature Value - the contents of address 0xA2 offset 96-97

According to SFF-8636, the Warning Threshold is optional, but since there is no indication about it, it will be reported as is and it's up to the BMC whether to use this information and how to interpret it.

10.6.4.19 Initialization Error AEN (Intel AEN 0x82)

Note: This AEN was previously called NVM error AEN.

The following is the AEN that might be sent by the NC following a detection of an error as part of the firmware initialization (reflected by a non null error in *GL_MNG_FWSM*). NC is required to store this AEN internally until a connection to the MC is established so the report could be issued in all cases.

This AEN must be enabled using the NC-SI AEN Enable command, using bit 18 (0x40000) of the AEN enable mask.

Bits				
Bytes	31:24	23:16	15:8	7:0
0...3	NC-SI AEN Header			
4...7				
8...11				
12...15				
16...19	Reserved			0x82
20	Index of Firmware module on which the error was found. The encoding is as defined in <i>GL_MNG_FWSM.EXT_ERR_IND</i> field.			

10.6.5 Basic NC-SI Workflows

10.6.5.1 Package States

A NC package can be in one of the following two states:

1. Selected — The package is allowed to use the NC-SI lines, meaning the NC package might send data to the MC.
2. De-selected — The package is not allowed to use the NC-SI lines, meaning, the NC package cannot send data to the MC.

The MC must select no more than one NC package at any given time. Package selection can be accomplished in one of two methods:

1. Select Package command — This command explicitly selects the NC package.
2. Any other command targeted to a channel in the package also implicitly selects that NC package.

Package de-select can be accomplished only by issuing the De-Select Package command. The MC should always issue the Select Package command as the first command to the package before issuing channel-specific commands. For further details on package selection, refer to the NC-SI specification.

10.6.5.2 Channel States

A NC channel can be in one of the following states:

1. Initial State — The channel only accepts the Clear Initial State command (the package also accepts the Select Package and De-Select Package commands).
2. Active State — This is the normal operational mode. All commands are accepted.

For normal operation mode, the MC should always send the Clear Initial State command as the first command to the channel.

10.6.5.3 Discovery

After interface power-up, the MC should perform a discovery process to discover the NCs that are connected to it. This process should include an algorithm similar to the following:

1. For package_id=0x0 to MAX_PACKAGE_ID:
 - a. Issue a Select Package command to package ID package_id.
 - b. If a response was received:
 - c. For internal_channel_id = 0x0 to MAX_INTERNAL_CHANNEL_ID.
 - d. Issue a Clear Initial State command for package_id | internal_channel_id (the combination of package_id and internal_channel_id to create the channel ID).
 - e. If a response was received:
 - f. Consider internal_channel_id as a valid channel for the package_id package.
 - g. The MC can now optionally discover channel capabilities and version ID for the channel.
 - h. Else, if a response was not received, then issue a Clear Initial State command three times.
 - i. Issue a De-Select Package command to the package (and continue to the next package).
 - j. Else, if a response was not received, issue a Select Packet command three times.

10.6.5.4 Configurations

This section details different configurations that should be performed by the MC.

It is good practice that the MC not consider any configuration valid unless the MC has explicitly configured it after every reset (entry into the initial state). As a result, it is recommended that the MC re-configure everything at power-up and channel/package resets.

10.6.5.4.1 NC Capabilities Advertisement

NC-SI defines the Get Capabilities command. It is recommended that the MC use this command and verify that the capabilities match its requirements before performing any configurations. For example, the MC should verify that the NC supports a specific AEN before enabling it.

10.6.5.4.2 Receive Filtering

In order to receive traffic, the BMC must configure the NC with receive filtering rules. These rules are checked on every packet received on the LAN interface (such as from the network). Only if the rules matched, will the packet be forwarded to the BMC.

10.6.5.4.2.1 MAC Address Filtering

NC-SI defines three types of MAC address filters: unicast, multicast and broadcast. To be received (not dropped) a packet must match at least one of these filters. The MC should set one MAC address using the Set MAC Address command and enable broadcast and global multicast filtering.

Unicast/Exact Match (Set MAC Address command)

This filter filters on specific 48-bit MAC addresses. The MC must configure this filter with a dedicated MAC address.

The NC might expose three types of unicast/exact match filters (such as MAC filters that match on the entire 48 bits of the MAC address): unicast, multicast and mixed. The E610 exposes two mixed filters, which might be used both for unicast and multicast filtering. The MC should use one mixed filter for its MAC address.

Refer to NC-SI specification — Set MAC Address for further details.

Broadcast (Enable/Disable Broadcast Filter command)

NC-SI defines a broadcast filtering mechanism that has the following states:

1. Enabled — All broadcast traffic is blocked (not forwarded) to the BMC, except for specific filters (such as ARP request, DHCP, and NetBIOS).
2. Disabled — All broadcast traffic is forwarded to the BMC, with no exceptions.

Refer to NC-SI specification Enable/Disable Broadcast Filter command.

Global Multicast (Enable/Disable Global Multicast Filter)

NC-SI defines a multicast filtering mechanism which has the following states:

1. Enabled — All multicast traffic is blocked (not forwarded) to the BMC.
2. Disabled — All multicast traffic is forwarded to the BMC, with no exceptions.

The recommended operational mode is Enabled, with specific filters set. Not all multicast filtering modes are necessarily supported. Refer to NC-SI specification Enable/Disable Global Multicast Filter command for further details.

10.6.5.4.3 VLAN

NC-SI defines the following VLAN work modes:

Mode	Command and Name	Descriptions
Disabled	Disable VLAN command	In this mode, no VLAN frames are received.
Enabled #1	Enable VLAN command with VLAN only	In this mode, only packets that matched a VLAN filter are forwarded to the MC.
Enabled #2	Enable VLAN command with VLAN only + non-VLAN	In this mode, packets from mode 1 + non-VLAN packets are forwarded.
Enabled #3	Enable VLAN command with Any-VLAN + non-VLAN	In this mode, packets are forwarded regardless of their VLAN state.

Refer to NC-SI specification — Enable VLAN command for further details.

The E610 only supports modes #1 and #3. Recommendation:

1. Modes:
 - a. If VLAN is not required — Use the disabled mode.
 - b. If VLAN is required — Use the enabled #1 mode.
2. If enabling VLAN, The MC should also set the active VLAN ID filters using the NC-SI Set VLAN Filter command prior to setting the VLAN mode.

10.6.5.5 PT Traffic States

The MC has independent, separate controls for enablement states of the receive (from LAN) and of the transmit (to LAN) PT paths.

10.6.5.6 Channel Enable

This mode controls the state of the receive path:

1. Disabled — The channel does not pass any traffic from the network to the MC.
2. Enabled — The channel passes any traffic from the network (that matched the configured filters) to the MC.

This state also affects AENs: AENs is only sent in the enabled state.

The default state is disabled.

It is recommended that the MC complete all filtering configuration before enabling the channel.

10.6.5.7 Network Transmit Enable

This mode controls the state of the transmit path:

1. Disabled — the channel does not pass any traffic from the MC to the network.
2. Enabled — the channel passes any traffic from the MC (that matched the source MAC address filters) to the network.

The default state is disabled.

The NC filters PT packets according to their source MAC address. The NC tries to match that source MAC address to one of the MAC addresses configured by the Set MAC Address command. As a result, the MC should enable network transmit only after configuring the MAC address.

It is recommended that the MC complete all filtering configuration (especially MAC addresses) before enabling the network transmit.

This feature can be used for fail-over scenarios. See [Section 10.6.9.3](#).

10.6.6 Asynchronous Event Notifications (AENs)

AENs are unsolicited messages sent from the NC to the MC to report status changes (such as link change, operating system state change, etc.).

Recommendations:

- The MC firmware designer should use AENs. To do so, the designer must take into account the possibility that a NC-SI response frame (such as a frame with the NC-SI EtherType), arrives out-of-context (not immediately after a command, but rather after an out-of-context AEN).
- To enable AENs, the MC should first query which AENs are supported, using the Get Capabilities command, then enable desired AEN(s) using the Enable AEN command, and only then enable the channel using the Enable Channel command.

10.6.7 Querying Active Parameters

The MC can use the Get Parameters command to query the current status of the operational parameters.

10.6.8 Resets

In NC-SI there are two types of resets defined:

1. Synchronous entry into the initial state.
2. Asynchronous entry into the initial state.

Recommendations:

- It is very important that the MC firmware designer keep in mind that following any type of reset, all configurations are considered as lost and thus the MC must re-configure both the synchronous and asynchronous entries.
- As an asynchronous entry into the initial state might not be reported and/or explicitly noticed, the MC should periodically poll the NC with NC-SI commands (such as Get Version ID, Get Parameters, etc.) to verify that the channel is not in the initial state. Should the NC channel respond to the command with a Clear Initial State Command Expected reason code, the MC should consider the channel (and most probably the entire NC package) as if it underwent a (possibly unexpected) reset event. Thus, the MC should re-configure the NC. See the NC-SI specification section on Detecting Pass-through Traffic Interruption.
- The Intel recommended polling interval is 2-3 seconds.

For exact details on the resets, refer to NC-SI specification.

10.6.9 Advanced Workflows

10.6.9.1 Multi-NC Arbitration

As described in [Section 10.6.1.2](#), in a multi-NC environment, there is a need to arbitrate the NC-SI lines.

Figure 10-8 shows the system topology of such an environment.

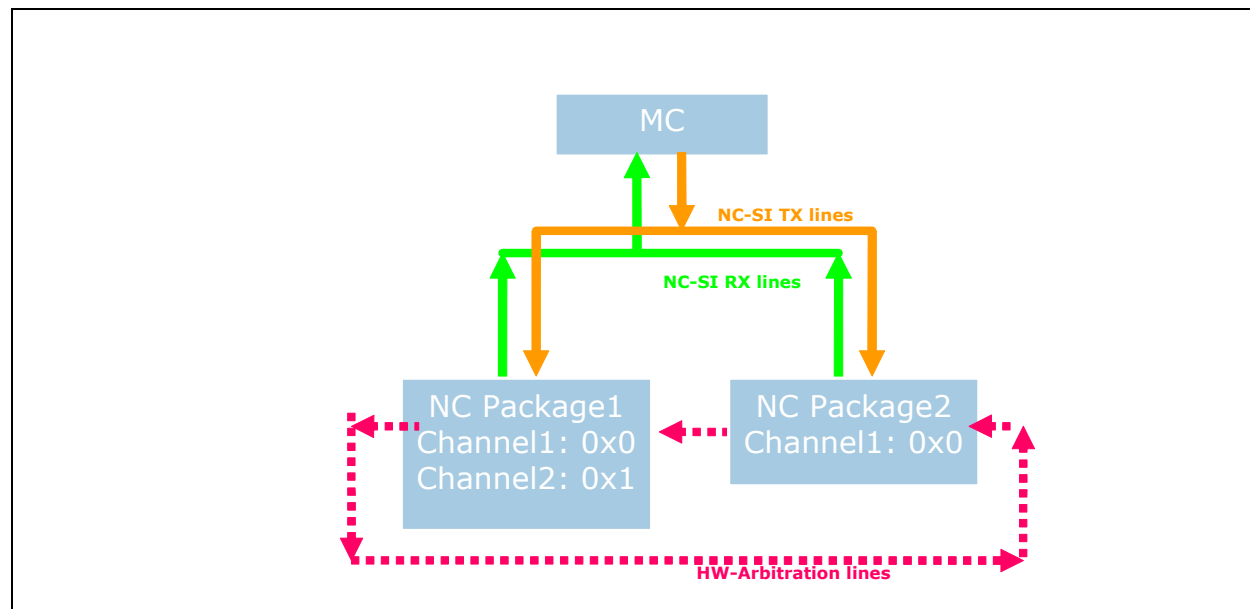


Figure 10-8. Multi-NC Environment

See [Figure 10-8](#). The NC-SI Rx lines are shared between the NCs. To enable sharing of the NC-SI Rx lines, NC-SI has defined an arbitration scheme.

The arbitration scheme mandates that only one NC package can use the NC-SI Rx lines at any given time. The NC package that is allowed to use these lines is defined as selected. All the other NC packages are de-selected.

NC-SI has defined two mechanisms for the arbitration scheme:

1. Package selection by the MC. In this mechanism, the MC is responsible for arbitrating between the packages by issuing NC-SI commands (Select/De-Select Package). The MC is responsible for having only one package selected at any given time.
2. Hardware arbitration. In this mechanism, two additional pins on each NC package are used to synchronize the NC package. Each NC package has an ARB_IN and ARB_OUT line and these lines are used to transfer tokens. A NC package that has a token is considered selected.

Note: Hardware arbitration is enabled by the NC-SI *HW Arbitration Enable* configuration bit in the NC-SI Configuration 1 NVM word.

For details, refer to the NC-SI specification.

10.6.9.2 Package Selection Sequence Example

Following is an example work flow for a MC and occurs after the discovery, initialization, and configuration.

Assuming the MC needs to share the NC-SI bus between packages, the MC should:

1. Define a time-slot for each device.
2. Discover, initialize, and configure all the NC packages and channels.
3. Issue a De-Select Package command to all the channels.
4. Set active_package to 0x0 (or the lowest existing package ID).
5. At the beginning of each time slot the MC should:
 - a. Issue a De-Select Package command to the active_package. The MC must then wait for a response and then an additional timeout for the package to become de-selected (200 μ s). See the NC-SI specification table 10 — parameter NC Deselect to Hi-Z Interval.
 - b. Find the next available package (typically active_package = active_package + 1).
 - c. Issue a Select Package command to active_package.

10.6.9.3 Multiple Channels (Fail-Over)

In order to support a fail-over scenario, it is required from the MC to operate two or more channels. These channels might or might not be in the same package.

The key element of a fault-tolerance fail-over scenario is having two (or more) channels identifying to the switch with the same MAC address, but only one of them being active at any given time (such as switching the MAC address between channels). To accomplish this, NC-SI provides the following commands:

1. Enable Network Tx command — This command enables shutting off the network transmit path of a specific channel. This enables the MC to configure all the participating channels with the same MAC address but only enable one of them.
2. Link Status Change AEN or Get Link Status command.

10.6.9.3.1 Fail-Over Algorithm Example

The following is a sample workflow for a fail-over scenario for the E610 (one package and four channels):

1. The MC initializes and configures all channels after power-up. However, the MC uses the same MAC address for all of the channels.
2. The MC queries the link status of all the participating channels. The MC should continuously monitor the link status of these channels. This can be accomplished by listening to AENs (if used) and/or periodically polling using the Get Link Status command.
3. The MC then only enables channel 0 for network transmission.
4. The MC then issues a gratuitous ARP (or any other packet with its source MAC address) to the network. This packet informs the switch that this specific MAC address is registered to channel 0's specific LAN port.
5. The MC begins normal workflow.

6. Should the MC receive an indication (AEN or polling) that the link status for the active channel (channel 0) has changed, the MC should:
 - a. Disable channel 0 for network transmission.
 - b. Check if a different channel is available (link is up).
 - c. If found:
 - Enable network Tx for that specific channel.
 - Issue a gratuitous ARP (or any other packet with its source MAC address) to the network. This packet informs the switch that this specific MAC address is registered to channel 0's specific LAN port.
 - Resume normal workflow.
 - If not found, report the error and continue polling until a valid channel is found.

The previous algorithm can be generalized such that the start-up and normal workflow are the same. In addition, the MC might need to use a specific channel (such as channel 0). In this case, the MC should switch the network transmit to that specific channel as soon as that channel becomes valid (link is up).

Recommendations:

- Wait for a link-down-tolerance timeout before a channel is considered invalid. For example, a link re-negotiation might take a few seconds (normally 2 to 3 or might be up to 9). Thus, the link must be re-established after a short time.
- Typically, this timeout is recommended to be three seconds.
- Even when enabling and using AENs, periodically poll the link status, as dropped AENs might not be detected.

10.6.9.4 Statistics

The MC might use the statistics commands as defined in NC-SI. These counters are intended for debug purposes and are not all supported.

The statistics are divided into three commands:

1. Controller statistics — These are statistics on the network interface (to the host operating system and the PT traffic). See the NC-SI specification for details.
2. NC-SI statistics — These are statistics on the NC-SI control frames (such as commands, responses, AENs, etc.). See the NC-SI specification for details.
3. NC-SI PT statistics — These are statistics on the NC-SI PT frames. See the NC-SI specification for details.

10.7 MCTP

10.7.1 Management Component Transport Protocol (MCTP) Overview

MCTP defines a communication model intended to facilitate communication between:

- MCs and other MCs
- MCs and management devices

The communication model includes a message format, transport description, message exchange patterns, and configuration and initialization messages.

The basic MCTP specification is described in DMTF's DSP0236 document.

MCTP is designed so that it can potentially be used on many bus types. The protocol is intended to be used for intercommunication between elements of platform management subsystems used in computer systems, and is suitable for use in mobile, desktop, workstation, and server platforms.

Currently, specifications exist for MCTP over PCIe (DMTF's DSP0238) and over SMBus (DMTF's DSP0237). A specification for MCTP over USB is also planned.

MCs such as a Baseboard Management Controller (BMC) can use this protocol for communication between one another, as well as for accessing management devices within the platform.

10.7.1.1 MCTP Usage Model

The E610 supports MCTP protocol over the PCIe and SMBus buses. The E610 can connect through MCTP to a MC or the ME engine in the chipset as described in [Figure 10-9](#).

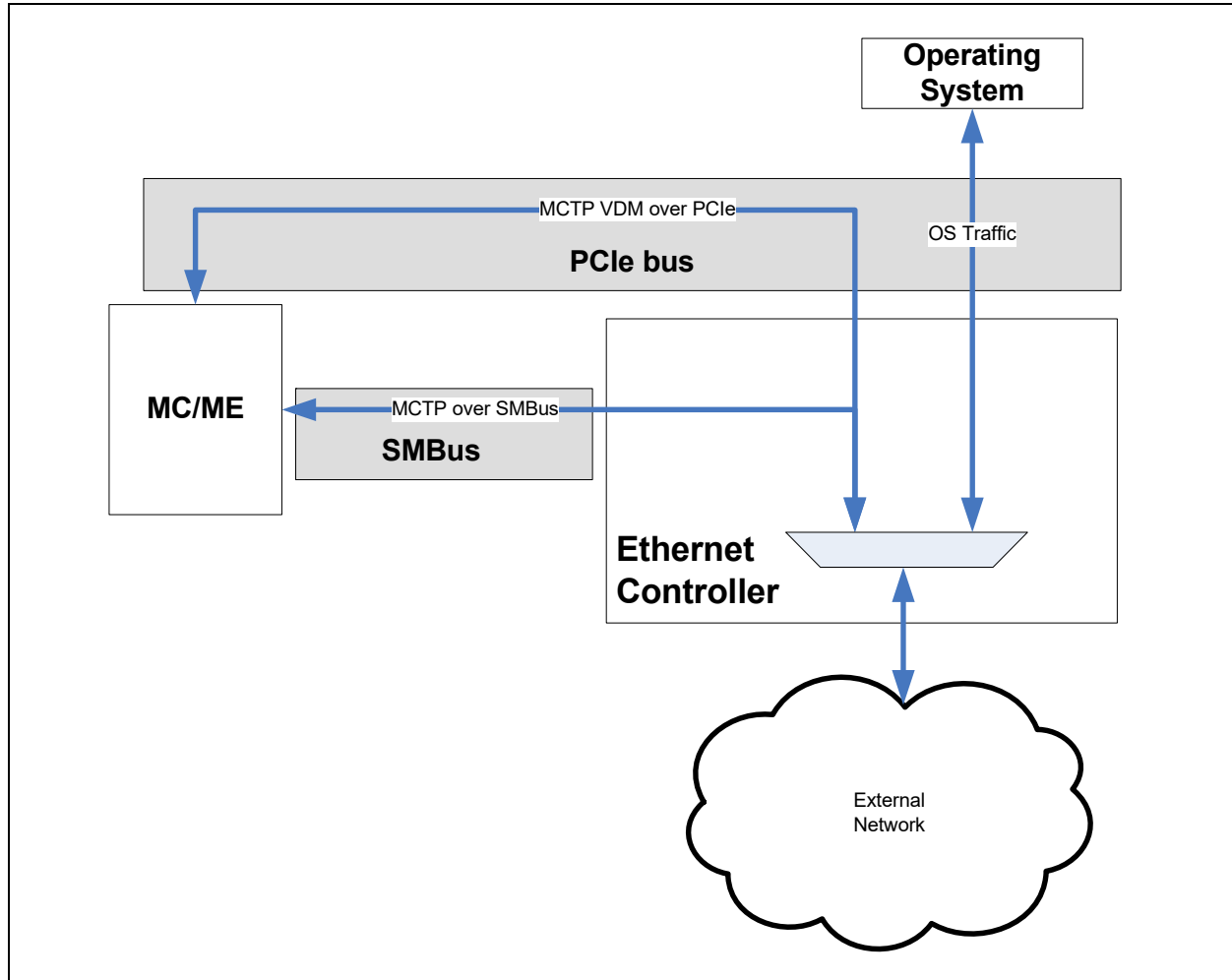


Figure 10-9. the E610 MCTP Connections

10.7.1.2 Detecting an MC EID and Physical Address

In order to enable transactions between the MC and the NIC, the bus physical address (SMBus or PCIe) and the EID of the partner needs to be discovered. NICs don't try to discover the MC and assume the MC initiates the connection. If the NIC is in an NC-SI initial state, then the EID and the physical address of the MC are extracted from the Clear Initial State command parameters or any other NC-SI command received later with a channel ID of the E610. Subsequent PT traffic is received from or sent to this address only.

If the EID or the physical address of the NIC changes, it indicates the changes to bus owner so that the routing tables can be updated. There is no attempt to directly send an indication to the MC about the change.

See more details in next section.

10.7.1.3 Bus Transition

The following section defines the transition flow between PCIe and SMBus as the bus on which MCTP flows. Figure 10-10 describes the flow to transition between PCIe and SMBus. The following parameters are used to define the flow:

- NIC EID on PCIe
- NIC EID on SMBus
- NIC PCIe Target ID
- Bus Owner EID on PCIe
- Bus Owner EID on SMBus
- Bus Owner PCIe Target ID
- Bus Owner SMBus Address
- MC EID on PCIe
- MC EID on SMBus
- MC PCIe Target ID
- MC SMBus Address
- NIC SMBus Address

All these variables are initialized to zero at power on apart from the SMBus address of the endpoint (NIC), which might be initialized from a NVM value.

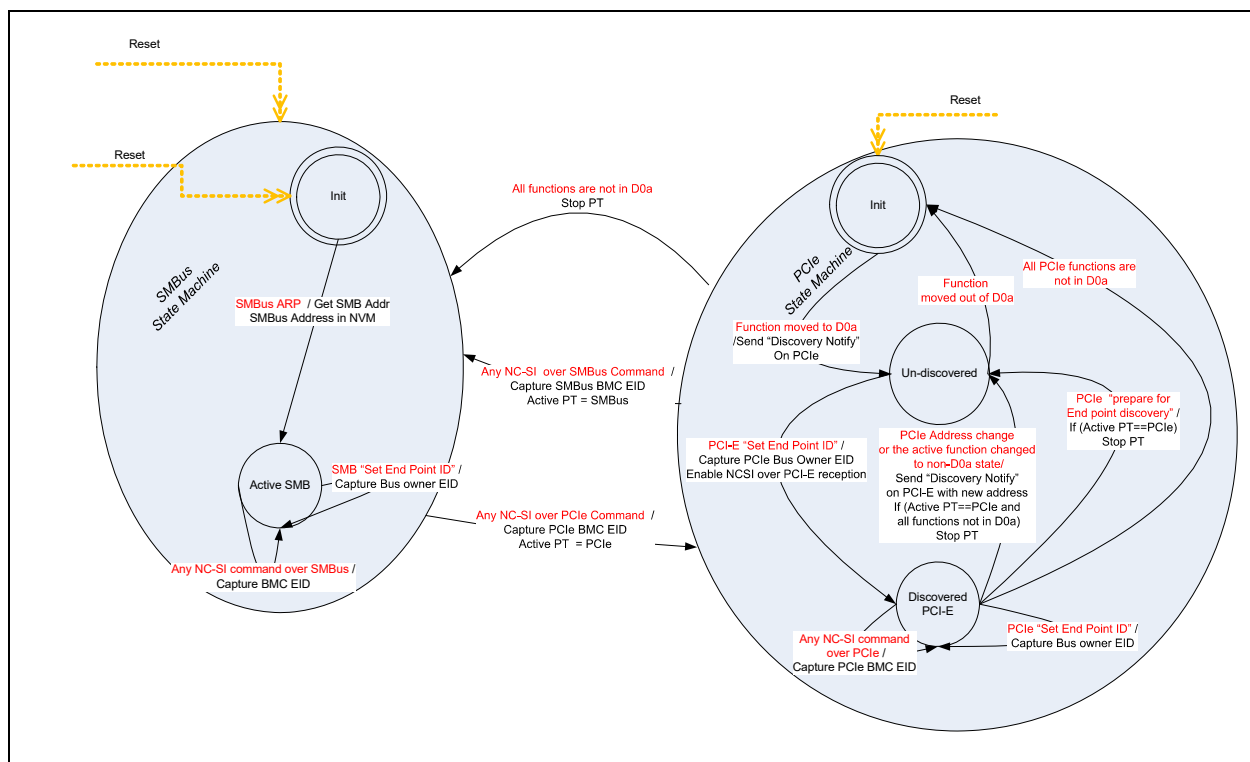


Figure 10-10. MCTP Bus Transition State Machine

10.7.1.4 Initial Assignment Flow

- At power on, the NIC or MC MCTP channel is connected to the SMBus, is not assigned an EID and is in an undiscovered state.
- The bus owner might preform an SMBus ARP cycle to assign an SMBus address to the NIC or to the MC. Otherwise, a fixed address might be used. It is assumed that the SMBus address does not change after initialization time.
- The bus owner performs an EID assignment using a Set Endpoint ID MCTP command. The NIC or the MC captures the SMBus address of the bus owner from the *SMBus Source Responder address* field, the bus owner EID from the *Source Endpoint ID* field and the NIC/MC EID from the *Destination Endpoint ID* field in the MCTP header as described in section 10.3 of DSP0236. The NIC/ MC is now in a discovered state.
- The MC might detect the NIC EID using one of the two following modes:
 - Static configuration of the NIC SMBus address in the MC database and Get Routing Table Entries command to find the EID matching the SMBus address.
 - Get all endpoints through a Get Routing Table Entries command and find endpoints supporting NC-SI using the Get Message Type Support command for each endpoint.
- Once the NIC is found, the MC might send a Clear Initial State command to the NIC to start the NC-SI configuration. The NIC captures the MC SMBus address and MC EID from any NC-SI command received.
- After the NC-SI channels are enabled, traffic might be sent using the MC and NIC addresses previously discovered.
- The MC might also send a Get UUID command to get a unique identifier of the NIC that might be used later for re-connection upon topology changes.

10.7.1.4.1 SMBus to PCIe Transition

- If the NIC or the MC detects that the PCIe bus is available by detecting a function that moved to D0 state, it might request a transition using a Discovery Notify MCTP command on the PCIe bus. This command should be sent with a route to root-complex addressing as described in DSP0238 section 6.8. The source EID should be the EID previously assigned on the SMBus.
- If the NIC or the MC detects that the PCIe bus is available by detecting a non-zero bus number as reflected in the *FUNC_RID.BUS_NUMBER* field, it might request a transition using a Discovery Notify MCTP command on the PCIe bus. This command should be sent with a route to root-complex addressing as described in DSP0238 section 6.8. The source EID should be the EID previously assigned on the SMBus.
- After receiving the Discovery Notify MCTP command on the PCIe bus, the bus owner sends a Set Endpoint ID MCTP command on the PCIe bus and updates the routing table. The bus owner might choose to wait for the Discovery Notify MCTP command of both the MC and the NIC to do the transition. The bus owner should try to keep the EID previously assigned on the SMBus as the EID on PCIe bus.
- After receiving the Set Endpoint ID MCTP command, the NIC waits for an NC-SI command from the MC indicating it is ready to transition the connection to PCIe. After receiving such a command, the NIC transitions its PT traffic to the PCIe bus using the newly received addresses.

- The MC on its side, needs to discover the PCIe address of the NIC. This can be done using the Resolve Endpoint ID command if only the physical address changed or using the Resolve Endpoint UUID command also if both EID and physical address changed. It can then send an NC-SI command to the NIC to initiate the transition. The MC should not send any PT packets from the moment it sent the first NC-SI command on the PCIe and the moment a response is received for this command.
- The transition of NC-SI traffic (PT or commands/responses) from SMBus to PCIe should be done on a packet boundary and should not interrupt a packet fragmentation or reassembly.

10.7.1.4.2 PCIe Target ID Change

The target ID of one of the endpoints might change due to a new enumeration of the PCIe bus. In this case the following flow should be used:

The target ID of one of the endpoints might change, either due to a new enumeration of the PCIe bus or due to the disabling of one of the functions in the device (move to a non D0 state). In this case, the following flow should be used:

- The endpoint should send a Discovery Notify MCTP command on the PCIe bus using the new Requester ID.
- After receiving the Discovery Notify MCTP command with the new requester ID, the bus owner sends a Set Endpoint ID MCTP command on the PCIe bus and updates the routing table. The bus owner should try to keep the EID previously assigned on the SMBus as the EID on the previous requester ID.
- The bus owner sends an Routing Information Update command to all supporting endpoints that might then update the parameters of their counterpart they use.

10.7.1.4.3 PCIe to SMBus Transition

- If the NIC or the MC detects that the PCIe bus is not available by detecting a transition of all functions to a non D0 state, it stops using the PCIe for PT traffic or NC-SI traffic.
- If the NIC or the MC detects that the PCIe bus is not available by detecting a transition to Dr state, it stops using the PCIe for PT traffic or NC-SI traffic.
- Upon detection of the unavailability of the PCIe bus, the MC transitions the NC-SI channel to the MCTP over SMBus as previously described.
- The transition of NC-SI traffic (PT or commands/responses) from PCIe to SMBus might done at any stage and might interrupt a packet fragmentation or reassembly, as it is assumed that such a transition occurs only when the PCIe bus is not available anymore.

10.7.2 MCTP over PCIe

10.7.2.1 Message format

The message format used for NC-SI over MCTP over PCIe is as follows:

PCIe TLP header
MCTP header
NC-SI header and payload

Table 10-27. NC-SI/Ethernet over MCTP over PCIe Message Format

+0								+1								+2								+3								
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	
FMT 011		Type 10r2r1r0 ¹						R	TC 000		R	A t t ² _r	R	T H 2	T D 2	E P 2	Attr [1:0] 2		AT 00		Length 00_000x_xxxx											
PCI Requester ID															PCI Tag Field						Message Code Vendor Defined = 0111_1111b											
															R	Pad Len		MCTP VDM code - 0000b														
PCI Target ID (For Route by ID messages, otherwise = Reserved)															Vendor ID = 0x1AB4 (DMTF)																	
MCTP Reserved		Header version = 1			Destination endpoint ID										Source endpoint ID						S O M	E O M	SEQ#		T O	Tag						
I C	Message Type = 0x02/ 0x03						NC-SI Command/Pass Through data																									
.....																																
NC-SI Command/Pass Through data																																

1. r2r1r0 =
000b: Route to Root Complex.
010b: Route by ID.
011b: Broadcast from Root Complex.
2. TD = 0, EP = 0, TH = 0, Attr[2:0] = 0 for sent packets and is ignored for received packets.

10.7.2.2 PCIe Discovery Process

The E610 follows the discovery process described in section 6.9 of the MCTP PCIe VDM Transport Binding Specification (DSP0238).

After receiving an endpoint discovery message (while in undiscovered stage), the E610 exposes the endpoint on the selected function as previously described.

If the selected function moves to D3 after the endpoint was discovered, or if the bus number of the E610 changes due to a re-enumeration of the bus, the E610 sends a discovery notify message to indicate to the MC that it should do a re-enumeration of the device to discover the new endpoint.

Note: E610 keeps track of the “current active function” - typically the first PF to be moved do the D0a state. If PF0 is active, and you move it to D3hot state, PF1 becomes the active function. PF1 remains the active function even if PF0 is returned to the D0a state. The E610 only handles Route-by-ID messages targeted to the active PF.

10.7.2.3 MCTP over PCIe Special Feature

The E610 supports Rate limiting (optional) feature of MCTP when running over PCIe:

10.7.2.3.1 MCTP Uplink Rate Limiting

As the PCIe link can carry a traffic bandwidth much higher than what the MC can sustain, in order to avoid drop of packets, the E610 allows rate limiting of the MCTP PT traffic. The E610 supports rate limiting between 1 Mb/s and 1 Gb/s. The following parameters define the behavior of the rate limiter:

- Max rate limit (fixed from NVM via the MCTP rate in the MCTP Rate Limiter Config 1 word).
- The max burst size (fixed from NVM via the *MCTP max credits* field in the in the MCTP Rate Limiter Config 2 word). To limit the max burst to one VDM, set this parameter to 5.
- Decision point (fixed from NVM via the *decision point* field in the in the MCTP Rate Limiter Config 2 word).

This feature can be controlled dynamically by the DSP0236 Update rate limit command.

10.7.3 MCTP Over SMBus

The message format used for NC-SI over MCTP over SMBus is as follows:

SMBus header/PEC
MCTP header
NC-SI header and payload

Table 10-28. NC-SI/Ethernet over MCTP over SMBus Message Format

+0								+1								+2								+3							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Destination Responder Address							0	Command Code = MCTP = 0Fh								Byte count								Source Responder Address							1
MCTP Reserved			Header version = 1					Destination endpoint ID								Source endpoint ID								SOM	EOM	SEQ#		TO	Tag		
ICI	Message Type = 0x02/0x03							NC-SI Command/Pass Through data																							
.....																															
NC-SI Command/Pass Through data																															
PEC																															

1. IC = 0 for pass through sent packets, or as defined by firmware for other packets. Packet received with IC = 1 handling can be defined per message type.

10.7.3.1 SMBus Discovery Process

The E610 follows the discovery process described in section 6.5 of the MCTP SMBus/I²C Transport Binding Specification (DSP0237). It indicates support for ASF in the SMBus getUID command (see [Section 10.3.2.1.3.2](#)). It responds to any SMBus command using the MCTP command code. This ensures that the bus owner knows the E610 supports MCTP.

Note: MCTP commands over SMBus are received from any initiator address and are answered to the sender. There is no capturing of the bus owner address from any specific command.

10.7.3.2 MCTP over SMBus special features

The E610 supports the following optional feature of MCTP when running over SMBus:

- Fairness arbitration

10.7.3.2.1 Fairness Arbitration

When sending MCTP messages over SMBus and when fairness arbitration is enabled (see [Section 6.3.50.3](#)), the E610 should adhere to the fairness arbitration as defined in section 5.13 of DSP0237 when sending MCTP messages.

10.7.4 NC-SI over MCTP

MCTP is a transport layer protocol that does not include the functionality required to control the PT traffic required for an MC connection to the network. This functionality is provided by encapsulating NC-SI traffic as defined in DMTF's DSP0222 document.

The details of NC-SI over MCTP protocol are defined in the DMTF's DSP0261 - NC-SI Over MCTP Specification.

An NC-SI over MCTP implementation guide can be found in the DMTF's DSP0219 white paper.

The NC-SI over MCTP specification defines two types of MCTP message types: NC-SI (0x2) and Ethernet (0x3). The E610 supports both messages. When used only for control, then only the NC-SI (0x2) message type is supported.

In addition to the previous message types supported by the E610, the PCIe based VDM message type is also supported over PCIe to support ACL commands.

Details of the NC-SI over MCTP can be found in [Section 10.7.4](#).

Enabling NC-SI over MCTP for pass through traffic is done by setting the *Redirection Sideband Interface* field in the *Common Manageability Parameters* NVM word to "MCTP over PCI and SMBus".

Enabling NC-SI over MCTP for control traffic is done by setting the *Control Interface* field in the *Common Manageability Parameters* NVM word to "MCTP over SMBus/PCI" and by setting the *NC-SI* bit in *Common Manageability Parameters 2* NVM word.

The E610 support for NC-SI over MCTP is similar to the support for NC-SI over RBT with the following exceptions:

1. A set of new NC-SI OEM commands used to expose the NC-SI over MCTP capabilities.
2. The format of the packets is modified to account for the new transport layer as described in the sections that follow.

10.7.4.1 NC-SI to MCTP Mapping

The eight network ports of the E610 (mapped to eight NC-SI channels) are mapped to a single MCTP endpoint on SMBus and to another endpoint over PCIe.

The PCIe endpoint is mapped to a PCIe requester ID according to the following flow:

1. If the *Bus initiator Enable* bit of at least one of the functions is set, the endpoint is mapped to function the first available function.
2. If the *Bus initiator Enable* bits of all functions are cleared, the MCTP endpoint on PCIe is not exposed and the MCTP traffic is routed through the SMBus endpoint.

The responder address used for the SMBus endpoint is the responder address of the first port.

[Section 10.7.1.2](#) describes the transition between the two buses.

Both endpoints (SMBus and PCIe) might be active concurrently. However, PT traffic might be transferred only through one of them. If the PCIe endpoint is active, it is used for PT traffic; otherwise, the SMBus endpoint is used. The Set EID command can be used to force the transition for the PCIe endpoint to the SMBus endpoint if the bus owner determines the PCIe channel is not functional.

For each channel (SMBus or PCIe), the E610 should expect MCTP commands from two sources: the bus owner and the MC. In addition, it should expect PT traffic through one interface only. Thus, it should be able to process up to five interleaved commands/data:

- An MCTP control/OEM command from the PCIe bus owner (single packet message).
- An MCTP control/OEM command from the SMBus bus owner (single packet message).
- An MCTP control/OEM command from the MC over SMBus (single packet message).
- An MCTP control/OEM command from the MC over PCIe (single packet message).
- An NC-SI command or Ethernet packet from the MC over the active channel.

A single source should not interleave packets it sends.

The topology used for MCTP connection is shown in [Figure 10-11](#).

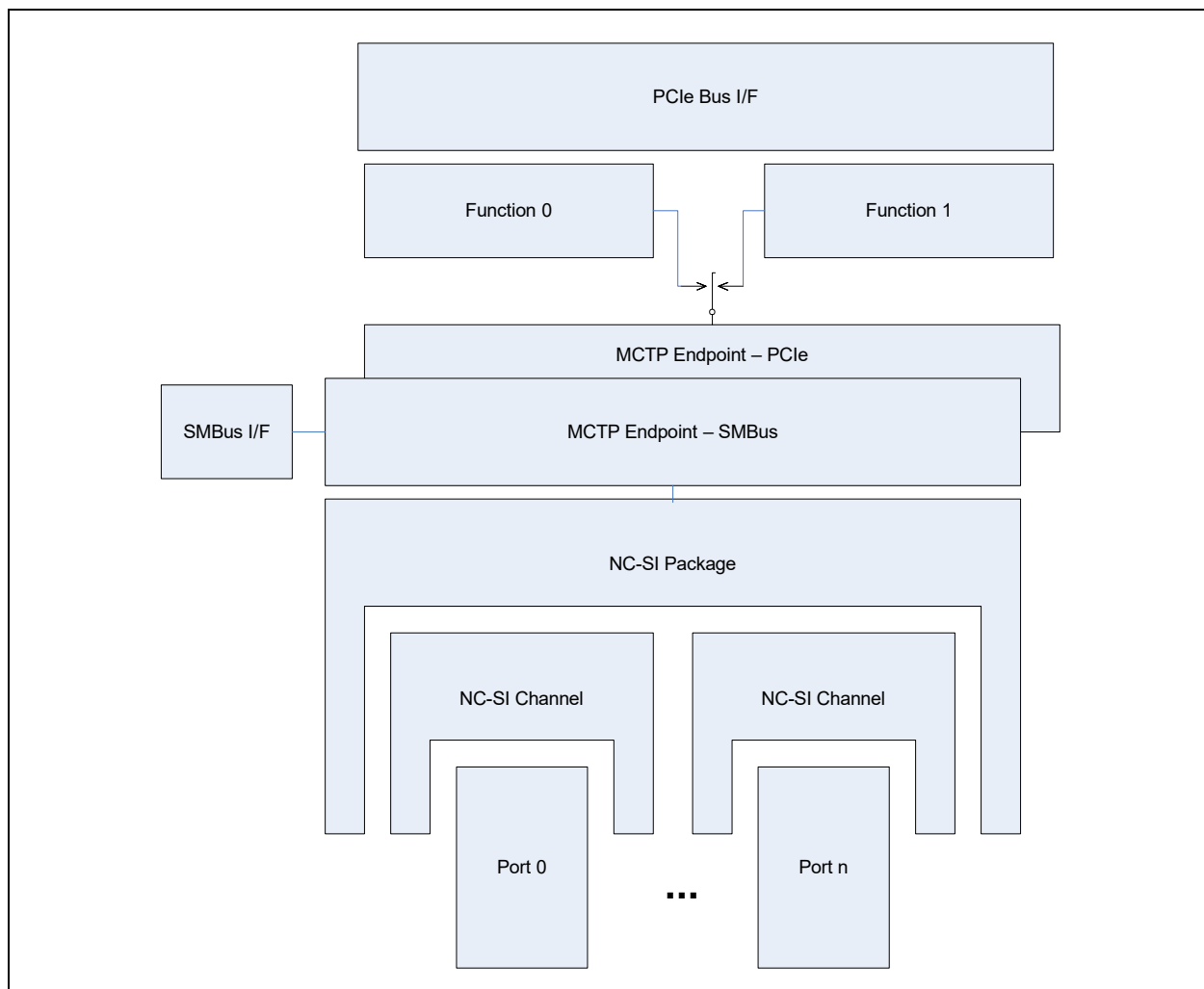


Figure 10-11. MCTP Endpoints Topology

10.7.4.2 NC-SI Packets Format

NC-SI over MCTP defines two different message type for pass through and for control packets.

Packets with a message type equal to the *Control packets message type* field (default = 0x02) in the NVM are NC-SI control packets (commands, responses and AENs) and packets with a message type equal to the *Pass through packets message type* field (default = 0x03) in the NVM are NC-SI pass through packets

10.7.4.2.1 Control Packets

The format used for control packets (commands, responses and AENs) is as follows:

Table 10-29. NC-SI over MCTP over PCIe/SMBus message format

+0								+1								+2								+3								
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	
SMBus or PCIe header																																
MCTP Reserved				Header version = 1				Destination endpoint ID								Source endpoint ID								SOM	EOM	SEQ#		TO = 1	Tag			
IC = 0	Message Type = Control Packets Message type (0x02)							MC ID = 0x00								Header revision								Reserved								
IID								Command								Channel ID ¹								Reserved				Payload Length[11:8]				
Payload Length[7:0]								Reserved																								
Reserved																																
Reserved								Command Data																								
Command Data																																
Command Data																Checksum																
Checksum																																

1. The channel ID is defined as described in [Section 10.3.2.2](#).

SMBus/PCIe header
MCTP header
NC-SI header
NC-SI Data

Note that the MAC header and MAC FCS present when working over NC-SI are not part of the packet in MCTP mode.

10.7.4.2.2 Pass-Through Packets

The format used for Pass Through packets are as follows. This format is the same for either packets received from the network or packets received from the host.

The CRC is never included in the packet. In receive, the CRC is checked and removed by the E610 in transmit, the CRC is added by the E610.

Table 10-30. Ethernet over MCTP over PCIe/SMBus Message Format

+0								+1								+2								+3								
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	
SMBus or PCIe header																																
MCTP Reserved				Header version = 1				Destination endpoint ID								Source endpoint ID								SOM	EOM	SEQ#		TO = 1	Tag			
IC = 0	Message Type = Pass Through Packets Control Type						DA																									
	DA																SA															
	SA																															
	SA						Ether type																Ethernet Packet									
	Ethernet packet																															
....																																
....																																

10.7.5 PLDM over MCTP

Enabling PLDM over MCTP for control traffic is done by setting the *Control Interface* field in the *Common Manageability Parameters* NVM word to “MCTP over SMBus/PCI” and by setting the *PLDM* bit in *Common Manageability Parameters 2* NVM word.

10.7.6 OEM Commands over MCTP

Enabling OEM commands over MCTP for control traffic is done by setting the *Control Interface* field in the *Common Manageability Parameters* NVM word to “MCTP over SMBus/PCI” and by setting the *OEM commands* bit in *Common Manageability Parameters 2* NVM word.

10.7.7 MCTP Programming

The MCTP programming model is based on:

1. A set of MCTP commands used for the discovery process and for the link management. The list of supported commands is described in section [Section 10.7.7.1](#).
2. A subset of the NC-SI commands used in the regular NC-SI interface, including all the OEM commands as described in [Section 10.6.2](#) (NC-SI programming I/F). The specific commands supported are listed in [Table 10-11](#) and [Table 10-14](#).

Note: For all MCTP commands (both native MCTP commands and NCSI over MCTP), the response uses the Msg tag received in the request with *TO* bit cleared.

10.7.7.1 MCTP Commands Support

[Table 10-31](#) lists the MCTP commands supported by the E610.

Table 10-31. MCTP Commands Support

Command Code	Command Name	General Description	E610 Support as Initiator	E610 Support as Responder
0x00	Reserved	Reserved	–	–
0x01	Set Endpoint ID	Assigns an EID to the endpoint at the given physical address.	N/A	Yes
0x02	Get Endpoint ID	Returns the EID presently assigned to an endpoint. Also returns information about what type the endpoint is and its level of use of static EIDs. See Section 10.7.7.1.1 for details.	No	Yes
0x03	Get Endpoint UUID	Retrieves a per-device unique UUID associated with the endpoint. See Section 10.7.7.1.2 for details.	No	Yes
0x04	Get MCTP Version Support	Lists which versions of the MCTP control protocol are supported on an endpoint. See Section 10.7.7.1.3 for details.	No	Yes
0x05	Get Message Type Support	Lists the message types that an endpoint supports. See Section 10.7.7.1.4 for details.	No	Yes
0x06	Get Vendor Defined Message Support	Used to discover an MCTP endpoint's vendor specific MCTP extensions and capabilities. See Section 10.7.7.1.5 for details.	No	Yes
0x07	Resolve Endpoint ID	Used to get the physical address associated with a given EID.	Yes	N/A
0x08	Allocate Endpoint IDs	Used by the bus owner to allocate a pool of EIDs to an MCTP bridge.	N/A	N/A
0x09	Routing Information Update	Used by the bus owner to extend or update the routing information that is maintained by an MCTP bridge.	N/A	N/A
0x0A	Get Routing Table Entries	Used to request an MCTP bridge to return data corresponding to its present routing table entries.	No	N/A
0x0B	Prepare for Endpoint Discovery	Used to direct endpoints to clear their discovered flags to enable them to respond to the Endpoint Discovery command.	N/A	Yes ¹
0x0C	Endpoint Discovery	Used to discover MCTP-capable devices on a bus, provided that another discovery mechanism is not defined for the particular physical medium.	No	Yes ¹
0x0D	Discovery Notify	Used to notify the bus owner that an MCTP device has become available on the bus.	Yes ¹	N/A
0x0E	Get Network ID	Used to get the MCTP network ID.	No	No

Table 10-31. MCTP Commands Support [continued]

Command Code	Command Name	General Description	E610 Support as Initiator	E610 Support as Responder
0x0F	Query Hop	Used to discover what bridges, if any, are in the path to a given target endpoint and what transmission unit sizes the bridges will pass for a given message type when routing to the target endpoint.	No	No
0x11	Query rate limit	Used to discover the data rate limit settings of the given target for incoming messages.	No	Yes
0x12	Request TX rate limit	Used to request the allowed transmit data rate limit for the given endpoint for outgoing messages.	No	Yes
0x13	Update Rate limit	Used to update the receiving side on change to the transmit data rate which was not requested by the receiver	No	Yes
0x14	Query supported interfaces	Used to discover the existing device MCTP interfaces.	No	Yes

1. These commands are supported only for MCTP over PCIe.

10.7.7.1.1 Get Endpoint ID

The get endpoint ID response of the E610 is listed in the following table:

Byte	Description	Value
1	Completion Code	
2	Endpoint ID	0x00 - EID not yet assigned. Otherwise - returns EID assigned using Set Endpoint ID command.
3	Endpoint Type	0x00 (Dynamic EID, simple endpoint).
4	Medium Specific	SMBus: 0x01 - Fairness arbitration protocol supported. PCIe: 0x00.

10.7.7.1.2 Get Endpoint UUID

The UUID returned is calculated according to the following function:

- Time Low = Read from MCTP UUID - Time Low LSB/MSB NVM words of Sideband Configuration Structure.
- Time mid = Read from MCTP UUID - Time Mid NVM word of Sideband Configuration Structure
- Time High and version = Read from MCTP UUID - Time High and version NVM word of Sideband Configuration Structure
- Clock Sec and Reserved = Read from MCTP UUID - Clock Seq NVM word of Sideband Configuration Structure
- Node = MAC Address as taken from the *GLPCI_SERL* and *GLPCI_SERH* registers.

10.7.7.1.3 Get MCTP Version Support

The following table lists the returned value according to the requested message type. The list of supported message types is based on the protocols enabled in the NVM and should be the same as the list reported in the Get message type support command.

Byte	Description	Message Type						
		0xFF(Base)	0x00 (Control Protocol Message)	0x01 (PLDM)	0x02 (NC-SI Over MCTP) ¹	0x03 (Ethernet) ₂	0x7E (PCIe Based VDM Messages) ₃	All Other Or Unsupporte d Messages
1	Completion Code	0x0						0x80
2	Version Number entry count	4	4	1	2	2	2	0
6:3	Version number entry	0xF1F0FF00 (1.0)	0xF1F0FF00 (1.0)	0xF1F0F000 (1.0.0)	0xF1F0F000 (1.0.0)	0xF1F0F000 (1.0.0)	0xF1F0FF00 (1.0)	0
10:7	Version number entry 2	0xF1F1F000 (1.1.0)	0xF1F1F000 (1.1.0)		0xF1F1F000 (1.1.0)	0xF1F1F000 (1.1.0)	0xF1F1F000 (1.1.0)	
14:11	Version number entry 3	0xF1F2F000 (1.2.0)	0xF1F2F000 (1.2.0)		0xF1F2F000 (1.2.0)	0xF1F2F000 (1.2.0)		
19:15	Version number entry 4	0xF1F3F000 (1.3.0)	0xF1F3F000 (1.3.0)					

1. If NC-SI is supported in the current configuration over this medium
2. If NC-SI pass-through is supported in the current configuration over this medium
3. If OEM messages are supported in the current configuration over this medium

10.7.7.1.4 Get Message Type Support Command

The get message type support response of the E610 is listed in the following table:

Byte	Description	Value
1	Completion Code	0x00.
2	MCTP Message Type Count	0x01/0x02/0x03/0x4 - The E610 supports up to four additional message types, depending on the mode of operation and the bus used.
3:6	List of Message Type Numbers	0x01 (PLDM over MCTP) - If PLDM is supported.
		0x02 (NC-SI over MCTP) - If NC-SI is supported.
		0x03 (Ethernet). If NC-SI and pass through are supported.
		0x05 (SPDM) and 0x06 (Secured Messages) if SPDM is supported
		0x7E (PCIe based VDM messages) - over PCIe. Over SMBus also if OEM commands are supported.

10.7.7.1.5 Get Vendor Defined Message Support Command

The get vendor defined message type support response of E610 is listed in the following table if vendor ID set selector equals 0x00:

Byte	Description	Value
1	Completion Code	0x00.
2	Vendor ID Set Selector	0xFF = No more capability sets.
3:5	Vendor ID	0x008086 (PCI ID indicator + Intel vendor ID).
6:7	Version	0x0100 (version 1.0).

10.7.7.1.6 Set Endpoint ID Command

The E610 supports the set EID and force EID operations defined in the Set Endpoint ID command. When operating over PCIe, the set discovered flag operation is also supported. As endpoints in the E610 can be set only through their own interface, set EID and force EID are equivalent. The Reset EID operation is not supported by the E610.

The Set Endpoint ID response of the E610 is described in the following table:

Byte	Description	Value
1	Completion Code	0x00.
2	Completion Status	[7:6] = 00b - Reserved.
		[5:4] = 00b - EID assignment accepted.
		[3:2] = 00b - Reserved.
		[1:0] = 00b - Device does not use an EID pool.
3	EID Setting	If the EID setting was accepted, this value matches the EID passed in the request. Otherwise, this value returns the present EID setting.
4	EID Pool Size	Always return a zero.

10.7.7.1.7 Query Rate Limit

When receiving a Query Rate Limit command over PCIe the device should respond with the following parameters:

Offset	Description	Value
2:5	Receive information: receive buffer size in bytes	Size of MCTP to manageability buffer in bytes (+ headers)
6:9	Receive Information: maximum receive data rate limit in packets	0x1DCD65 – reflects a rate of 1Gbps in 64 bytes packets
10:13	Transmit Rate limiter capabilities: Maximum supported rate limit	0x1DCD65 – reflects a rate of 1Gbps in 64 bytes packets
14:17	Transmit Rate limiter capabilities: Minimum supported rate limit	0x7A1 – reflects a rate of 1 Mbps in 64 bytes packets. Note – this should cover all products
18:20	Transmit Rate limiter capabilities: Maximum supported burst size.	Size of manageability to MCTP buffer in 64 bytes packets (+ headers)
21:23	Present Transmit Rate Limit Burst Setting	According to current setting. Initial value based on NVM configuration.
24:27	Present Setting: EID Maximal Transmit data rate limit	According to current setting. Initial value based on NVM configuration.
28.1	Transmit Rate limiting operation capability: 0b – Transmit Rate limiting on this EID is applied to requested and non requested messages together 1b – Transmit Rate limiting on this EID is applied only to non-requested messages	1
28.0	Rate limiting Support on EID	1

Note: When received over SMBus, bit 28.0 should be set to zero and bits 6:9 should also be set to zero. Other fields are don't care.

10.7.7.1.8 Request TX Rate Limit

10.7.7.1.8.1 Request TX Rate Limit Response

When receiving a Request TX rate limit command from the MC over PCIe, the device should set its rate limit according to the requested values. If received over SMBus, an ERROR_INVALID_DATA shall be returned.

The setting of the rate limit is done as described in section 3.5.4.5MCTP over PCI-E Rate Limiter Configuration of hardware/firmware interface document

10.7.7.1.8.2 Request TX Rate Limit Send

The device currently does not require a rate limit from the MC assuming the supported BW of 1Gbps is large enough to absorb any MC traffic.

10.7.7.1.9 Update Rate Limit

The device never sends this command and completes it without error, but ignores it when received.

10.7.7.1.10 Query Supported Interfaces

The 0x14 - Query supported interfaces command allows the MC or the bus owner to detect the available interfaces.

When receiving a Query supported interfaces command over PCIe or SMBus the device should respond with the following parameters.

Bytes 3:4 are returned if SMBus is available and bytes 5:6 are returned if PCIe is available.

Offset	Description	Value
2	Supported Interfaces Count	1 or 2 according to the available interfaces
3	First interface Type	0x01 if SMBus speed is 100KHz 0x04 if SMBus speed is 400KHz
4	First interface EID	The EID assigned to the device over SMBus
5	Second interface Type	0x0A for PCIe 2.1 device 0x0B for PCIe 3.0 device 0x0C for PCIe 4.0 device
6	Second interface EID	The EID assigned to the device over PCIe

10.8 PLDM Support

The E610 supports PLDM over MCTP protocol. Within PLDM, the following data types are supported:

- 0x0 - PLDM Messaging Control and Discovery
- 0x5 - PLDM Firmware Update
- 0x2 - PLDM Monitoring and Control
- 0x6 - PLDM Redfish to the Endpoint (RDE)

This section does not include details of the PLDM spec itself. This section contains information on the parts of PLDM to implement and the actual data needed to fill the PLDM responses. Following is a list of relevant specifications:

- DSP0240: Platform Level Data Model (PLDM) Base Specification
- DSP0241: Platform Level Data Model (PLDM) over MCTP Binding Specification
- DSP0245: Platform Level Data Model (PLDM) IDs and Codes Specification
- DSP0248: Platform Level Data Model (PLDM) for Platform Monitoring and Control Specification
- DSP0249: Platform Level Data Model (PLDM) State Set Specification
- DSP0267: Platform Level Data Model (PLDM) Firmware Update
- DSP0218: Platform Level Data Model (PLDM) for Redfish Device Enablement

10.8.1 PLDM Base Implementation

PLDM is supported over PCIe or over SMBus. The packet format is described in [Table 10-32](#):

Table 10-32. PLDM over MCTP over PCIe/SMBus Message Format

+0								+1								+2								+3							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
SMBus or PCIe Header																															
MCTP Reserved				Header Version = 1				Destination Endpoint ID								Source Endpoint ID								SOM	EOM	SEQ#		TO = 1	Tag		
IC = 0	Platform Level Data Model (PLDM) (0x01)							RQ	D	RSVD	Instance ID					Hdr Ver	PLDM Type					PLDM Command Code									
PLDM Completion Code ¹								PLDM Payload																							
....																															
Command Data																															

1. In response only. This byte is not present in commands.

The description of the PCIe header & MCTP header field is as defined in the spec. The values of the PLDM fields are described in [Table 10-33](#).

Table 10-33. PLDM Fields

Field	Size (Bits)	Description
IC	1	Integrity Check - Always zero. Appears only in first packet of message.
Message Type	7	1 = PLDM Appears only in first packet of message.
RQ, D	2	Request, Datagram D, RQ 00b – For PLDM response messages. 01b – For PLDM request messages. 10b – Reserved. 11b – For Unacknowledged PLDM request messages or asynchronous notifications.
Instance ID	5	Identifier of a command. Should be used in the response also. Note: The PLDM specification requires the device (terminus) to send commands to generate events, so the device should generate and track instance IDs for the events it sends to the Event Receiver.
Hdr Ver	2	Header Version - Should be 0.
PLDM Type	6	0 for PLDM control messages. 2 for Platform Monitoring and Control messages. 5 for PLDM Firmware Update messages.
PLDM Command Code	8	Command code.
PLDM Completion Code	8	Response code - Appears only in response. Should be according to the Table 4. "Generic PLDM Completion Codes (PLDM_BASE_CODES)" in the DSP0240 PLDM Base Specification or specific codes written in each command.

10.8.1.1 Reset Conditions

The PLDM state is reset when the MCTP layer below it is reset. For PCIe VDM, this is EMPR and PERST reset events.

10.8.1.2 PLDM Control Commands

The following messages (as defined in DSP0240) should be supported. For these messages, the PLDM type is zero.

Table 10-34. PLDM Control Commands

Command	Command Code	Description	Section Reference
SetTID	0x01	Sets the TID used in Platform Monitoring and Control messages.	---
GetTID	0x02	Gets the TID currently stored in the device.	---
GetPLDMVersion	0x03	Return supported versions for different PLDM types.	10.8.1.2.1
GetPLDMTypes	0x04	Return supported PLDM types.	10.8.1.2.2
GetPLDMCommands	0x05	Return supported PLDM commands per type.	10.8.1.2.3

10.8.1.2.1 GetPLDMVersion (0x3)

The GetPLDMVersion command returns the supported versions of PLDM types. The command and response are described in [Table 10-35](#).

The command is supported for PLDM Type 0 (PLDM Messaging Control and Discovery), Type 2 (PLDM for Platform Monitoring and Control), Type 5 (PLDM Firmware Update), and Type 6 (PLDM for RDE).

Table 10-35. GetPLDMVersion Command and Response

Byte	Type	Request Data
0:3	uint32	DataTransferHandle – ignored by the device
4	enum8	TransferOperationFlag This field is an operation flag that indicates whether this is the start of the transfer. Possible values: <pre>{ GetNextPart = 0x00, GetFirstPart = 0x01 }</pre> Accepted only if <i>GetFirstPart</i> .
5	uint8	PLDMType This field identifies the PLDM Type whose version information is being requested.
Byte	Type	Response Data
0	enum8	CompletionCode Possible values: <pre>{ PLDM_BASE_CODES, INVALID_DATA_TRANSFER_HANDLE = 0x80, INVALID_TRANSFER_OPERATION_FLAG = 0x81, INVALID_PLDM_TYPE_IN_REQUEST_DATA = 0x83 }</pre>
1:4	uint32	NextDataTransferHandle = 0x00 Not used as always returning a single message.
5	enum8	TransferFlag Return StartAndEnd = 0x05.
4	-	Portion of PLDMVersionData For Type = 0 – PLDM Messaging Control and Discovery – return 0xF1F0F000 (1.0.0) For Type = 2 – PLDM for Platform Monitoring and Control – return 0xF1F2F000 (1.2.0) For Type = 5 – PLDM for Firmware Update – return 0xF1F0F000 (1.0.0) For Type = 6 – Platform Level Data Model (PLDM) for Redfish Device Enablement – return 0xF1F1F000 (1.1.0)



10.8.1.2.2 GetPLDMType (0x4)

The GetPLDMType command returns the supported PLDM types.
The command and response are described in [Table 10-36](#).

Table 10-36. GetPLDMType Command and Response

Byte	Type	Request Data
-	-	None.
Byte	Type	Response Data
0	enum8	CompletionCode Possible values: {PLDM_BASE_CODES}
1:8	bitfield8[8]	PLDMType Each bit represents whether a given PLDM Type is supported: 0b = PLDM Type is not supported. 1b = PLDM Type is supported. For bitfield8[N], where N = 0 to 7: [7] = PLDM Type N*8+7 supported. [..] = ... [1] = PLDM Type N*8+1 supported. [0] = PLDM Type N*8+0 supported. Return 0x65 (types 0, 2, 5, and 6). Note: Type should be reported as supported only if enabled in NVM.

10.8.1.2.3 GetPLDMCommand (0x5)

The GetPLDMCommand command returns the supported command of PLDM per type.

The command and response are described in [Table 10-37](#).

Table 10-37. GetPLDMCommand Command and Response

Byte	Type	Request Data
0	uint8	PLDMType This field identifies the PLDM Type for which command support information is being requested. Supported for types 0, 2, 5, and 6. Otherwise, returns an INVALID_PLDM_TYPE_IN_REQUEST_DATA.
1:4	ver32	Version This field identifies the version for the specified PLDM Type. Supported values are: For Type = 0: 0xF1F0F000 (1.0.0) For Type = 2: 0xF1F2F000 (1.2.0) For Type = 5: 0xF1F0F000 (1.0.0) For Type = 6: 0xF1F1F000 (1.1.0) Otherwise returns an INVALID_PLDM_VERSION_IN_REQUEST_DATA.
Byte	Type	Response Data
0	enum8	CompletionCode Possible values: <pre>{ PLDM_BASE_CODES, INVALID_PLDM_TYPE_IN_REQUEST_DATA = 0x83, INVALID_PLDM_VERSION_IN_REQUEST_DATA = 0x84 }</pre>
1:8	bitfield8[32]	PLDMCommands (up to 256 commands supported for the specified PLDM Type) Each bit represents whether a given PLDM command is supported: 0b = PLDM command is not supported. 1b = PLDM command is supported. For bitfield8[N], where N = 0 to 31: [7] = PLDM Command N*8+7 Supported. [...] = ... [1] = PLDM Command N*8+1 Supported. [0] = PLDM Command N*8 Supported. For type 0, return 0x3E (commands 1,2,3,4,5) as described in this section. For type 2, return 0x00003_0000_0000_0003_000F_0038 (according to the supported commands in Section 10.8.4 (SetTID & GetTID are not included)). For type 5, return 0x0000_0000_0000_0000_3C19_0006 (according to the supported commands in Section 10.8.4). For type 6, return 0x0003_0000_007B_0BFE (according to the supported commands in Section 10.8.6.2). Note: The commands reported must be according to the firmware's actual support for the commands where NC is the responder.

10.8.2 PLDM Monitoring and Control Support

10.8.2.1 PLDM Monitoring and Control Supported Commands

The following messages, as defined in DSP0248, should be supported. For these messages, the PLDM type is two.

Table 10-38. PLDM Monitoring and Control Commands

Command Code	Command	Supported?	Section Reference
0x01	SetTID (see DSP0240)	Yes	---
0x02	GetTID (see DSP0240)	Yes	---
0x03	GetTerminusUID	Yes	10.8.2.1.1
0x04	SetEventReceiver	Yes	10.8.2.1.2
0x05	GetEventReceiver	Yes	10.8.2.1.3
0x0A	PlatformEventMessage	Yes	10.8.2.1.4
0x0B	PollForPlatformEventMessage	Yes	10.8.2.1.5
0x0C	EventMessageSupported	Yes	10.8.2.1.6
0x0D	EventMessageBufferSize	Yes	10.8.2.1.7
0x10	SetNumericSensorEnable	Yes	10.8.2.1.8
0x11	GetSensorReading	Yes	10.8.2.1.9
0x12	GetSensorThresholds	Yes	10.8.2.1.10
0x13	SetSensorThresholds	Yes	10.8.2.1.11
0x14	RestoreSensorThresholds	No	---
0x15	GetSensorHysteresis	Yes	10.8.2.1.12
0x16	SetSensorHysteresis	No	---
0x17	InitNumericSensor	No	---
0x20	SetStateSensorEnables	Yes	10.8.2.1.13
0x21	GetStateSensorReadings	Yes	10.8.2.1.14
0x22	InitStateSensor	No	---
0x30	SetNumericEffectorEnable	No	---
0x31	SetNumericEffectorValue	No	---
0x32	GetNumericEffectorValue	No	---
0x38	SetStateEffectorEnables	No	---
0x39	SetStateEffectorStates	No	---
0x3A	GetStateEffectorStates	No	---
0x40	GetPLDMEventLogInfo	No	---
0x41	EnablePLDMEventLogging	No	---
0x42	ClearPLDMEventLog	No	---
0x43	GetPLDMEventLogTimestamp	No	---
0x44	SetPLDMEventLogTimestamp	No	---

Table 10-38. PLDM Monitoring and Control Commands [continued]

Command Code	Command	Supported?	Section Reference
0x45	ReadPLDMEventLog	No	---
0x46	GetPLDMEventLogPolicyInfo	No	---
0x47	SetPLDMEventLogPolicy	No	---
0x48	FindPLDMEventLogEntry	No	---
0x50	GetPDRRepositoryInfo	Yes	10.8.2.1.15
0x51	GetPDR	Yes	10.8.2.1.16
0x52	FindPDR	No	---
0x53	GetPDRRepositorySignature	Yes	10.8.2.1.17
0x58	RunInitAgent	No	---

10.8.2.1.1 GetTerminusUID (0x3)

The GetTerminusUID command returns a UUID of the device. The command and response are described in [Table 10-39](#).

Table 10-39. GetTerminusUID Command and Response

Type	Request Data
---	None.
Type	Response Data
enum8	completionCode Value: {PLDM_BASE_CODES}
UUID	UUID Value

UUID format is described in [Table 10-40](#).

Table 10-40. UUID Format

Field	UUID Byte	MSB
time low	1	MSB
	2	
	3	
	4	
time mid	5	MSB
	6	
time high and version	7	MSB
	8	
clock seq high and reserved	9	
clock seq low	10	

Table 10-40. UUID Format [continued]

Field	UUID Byte	MSB
Node	11	
	12	
	13	
	14	
	15	
	16	

The UUID Value content should be the same as returned as part of the Get Endpoint UUID MCTP command.

10.8.2.1.2 SetEventReceiver (0x4)

The SetEventReceiver command is used to indicate to the device what the address of the Event Receiver is (usually the BMC). The command and response are described in Table 10-41.

Table 10-41. SetEventReceiver Command and Response

Type	Request Data
enum8	eventMessageGlobalEnable This value is used to enable or disable event message generation from the terminus. See DSP0248 for details. 0 = Disable 1 = Enable Async — Allows the device to work in event mode (send PlatformEventMessage command). 2 = Enable Polling — Allows the device to work in polling mode (react to PollForPlatformEventMessage commands).
enum8	transportProtocolType Only value supported if <i>eventMessageGlobalEnable</i> = enable, is 0x00 (MCTP), ignored otherwise. The command is rejected and an <i>INVALID_PROTOCOL_TYPE completionCode</i> returned if the <i>transportProtocolType</i> is incorrect.
uint8	eventReceiverAddressInfo Reflects the EID of the event receiver, and should be used to create events messages. Valid only if <i>eventMessageGlobalEnable</i> is set to enable (1b).
Type	Response Data
enum8	completionCode Possible values: { PLDM_BASE_CODES, INVALID_PROTOCOL_TYPE = 0x80 }

When this command is received and acknowledged, if the *eventReceiverAddressInfo* changed from what is currently registered in the firmware, the device should find the B,D,F (Bus, Device, Function) of the event receiver as follows:

- If the source EID in the packet is the same as the *eventReceiverAddressInfo*, use the requester ID in the TLP as the B,D,F of the receiver.
- Otherwise, send a Resolve Endpoint ID MCTP command to get the B,D,F of the Event Receiver.
- If the command fails four times, events are not sent.

10.8.2.1.3 GetEventReceiver (0x5)

The GetEventReceiver command is used to read from the device the address of the Event Receiver it is connected to. The command and response are described in [Table 10-42](#).

Table 10-42. GetEventReceiver Command and Response

Type	Request Data
---	None.
Type	Response Data
enum8	completionCode Value: {PLDM_BASE_CODES}
enum8	transportProtocolType Return 0x00 (MCTP)
uint8	eventReceiverAddress Returns the last value that was set using the SetEventReceiver command, or zero if no such command was received.

10.8.2.1.4 PlatformEventMessage (0xA)

The PlatformEventMessage command is a command generated by the device to notify the event receiver of new events. The device should implement a timeout and retry mechanism to ensure that the command is acknowledged by the event receiver.

The retry parameters as defined in *Table 5 - Timing Specifications for PLDM Messages* of DSP0240 should be set to PN1=2 (2 retries) and PT2 (Time-out waiting for a response)=2 seconds.

The destination EID and B,D,F of the command are received through the SetEventReceiver command and Resolve Endpoint ID MCTP command.

The following rules define when to send the message for each sensor:

- For Thermal Sensor and Thermal Trip events the message should be generated every time any of the thresholds is crossed in any direction.
- For Link State sensor events, the message should be generated every time the link state changes.
- For Link Speed sensor events, the message should be generated every time there is a link up event.
- For Health state sensor events, the message should be generated every time there is a change in the health state.
- For power consumption, there is no need to generate messages.
- For configuration change events the message should be generated every time the configuration change state changes (e.g., when PDR changed).
- For presence sensors, the message should be generated every time a cable presence status changes.

The command and response are described in [Table 10-43](#):

Table 10-43. PlatformEventMessage Command and Response

Type	Request Data
uint8	formatVersion 0x01
uint8	TID Terminus ID for the terminus that originated the event message as received from the SetTID command.
enum8	eventClass Value (one of the following): 0x00 = sensorEvent — One of the sensors had a status change. 0x02 = redfishTaskExecutedEvent — A long running task spawned by an RDE Operation task has completed execution. 0x03 = redfishMessageEvent — A Redfish event has occurred. 0x04 = pldmPDRRepositoryChgEvent — A change has been made to the repository.
var	eventData Event data based on the <i>eventClass</i> (see Section 10.8.2.2)
Type	Response Data
enum8	completionCode Possible values: { PLDM_BASE_CODES, UNSUPPORTED_EVENT_FORMAT_VERSION = 0x81 }
enum8	Status - can be ignored.

10.8.2.1.5 PollForPlatformEventMessage (0xB)

This command enables the Event Receiver to poll for events from a PLDM terminus and acknowledge the receipt of the event message. The SetEventReceiver command enables polling of event messages if the PLDM terminus supports this command.

Table 10-44. PollForPlatformEventMessage Command and Response

Type	Request Data
uint8	formatVersion Version of the event format (the format and definition of the following bytes): 0x01 for this specification.
enum8	TransferOperationFlag The operation flag that indicates whether this is the start of the transfer. Possible values: 0x00 = GetNextPart 0x01 = GetFirstPart 0x02 = AcknowledgementOnly
uint32	dataTransferHandle A handle that is used to identify a package data transfer. This handle is ignored by the responder when the <i>TransferOperationFlag</i> is set to GetFirstPart. If <i>TransferOperationFlag</i> is set to GetNextPart, <i>dataTransferHandle</i> should be equal to the <i>nextDataTransferHandle</i> reported in the previous transfer response. Otherwise return an ERROR_INVALID_DATA completion code.
uint16	eventIDToAcknowledge An event previously received that should be acknowledged. The MC uses the null value 0x0000 when requesting the first entry from the terminus' event queue. The MC uses the sentinel 0xFFFF when in the middle of a multi-part event transfer.

Table 10-44. PollForPlatformEventMessage Command and Response [continued]

Type	Response Data
enum8	completionCode Possible values: <pre>{ PLDM_BASE_CODES, UNSUPPORTED_EVENT_FORMAT_VERSION = 0x81, EVENT_ID_NOT_VALID = 0x82 }</pre>
uint8	TID
uint16	eventID The Event ID for the returned event in this response. The device assigns the Event ID to an event so the requester can send the acknowledgment on the subsequence invocation of this command. A value of 0x0000 is returned if the terminus internal event queue is empty. If <i>TransferOperationFlag</i> in the request message was set to AcknowledgementOnly and the event queue is non-empty, the terminus supplies sentinel value 0xFFFF for this field. <i>eventID</i> value generated by firmware is increased for each new event (starting from 1).
All of the following fields are returned only if <i>eventID</i> is different than 0x0000 or 0xFFFF.	
uint32	nextDataTransferHandle A handle that is used to identify the next portion of the transfer.
enum8	TransferFlag The transfer flag that indicates what part of the transfer that this response represents. Possible values: 0x1 = Start 0x2 = Middle 0x4 = End 0x5 = StartAndEnd
uint8	eventClass Values: pldmPDRRepositoryChgEvent —A change has been made to the repository. redfishTaskExecuteEvent — A long running task spawned by an RDE Operation has completed execution. redfishMessageEvent — A Redfish event has occurred. sensorEvent — Events related to PLDM numeric and state sensors.
uint32	eventDataSize Size in bytes of <i>eventData</i> .
var	eventData Event data based on the <i>eventClass</i> .
uint32	eventDataIntegrityChecksum 32-bit CRC for the entirety of event data (all parts concatenated together, excluding this checksum). This field is omitted for non-final chunks of multi-part event messages (TransferFlag ≠ End or StartAndEnd) in the transfer. The DataIntegrityChecksum is not split across multiple chunks. If appending the DataIntegrityChecksum would cause this request message to exceed the negotiated maximum transfer chunk size, the DataIntegrityChecksum is sent as the only data in another chunk. For this command, the CRC-32 algorithm with the polynomial $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ (same as the one used by IEEE 802.3) is used for the integrity checksum computation. The CRC computation involves processing a byte at a time with the least significant bit first.

10.8.2.1.6 EventMessageSupported (0xC)

Table 10-45. EventMessageSupported Command and Response

Type	Request Data
uint8	formatVersion Version of the event format (the format and definition of the following bytes): 0x01 for this specification version
Type	Response Data
enum8	completionCode Value: { PLDM_BASE_CODES, UNSUPPORTED_EVENT_FORMAT_VERSION = 0x81 }
enum8	synchronyConfiguration This value indicates the messaging style most recently configured via the SetEventReceiver command: Value: { NOT_CONFIGURED = 0x00, // SetEventReceiver command not received ASYNCHRONOUS_MESSAGING = 0x01, // Asynchronous messaging SYNCHRONOUS_MESSAGING = 0x02 // Poll-based messaging ASYNCHRONOUS_WITH_HEARTBEAT = 0x03 // Asynchronous messaging, heartbeat }
bitfield8	synchronyConfigurationSupported This value indicates the event messaging styles supported by the terminus. For each bit, a value of 1b indicates that the mode is supported. [7:4] - Reserved for future use [3] - Asynchronous messaging with heartbeat [2] - Synchronous (poll-based) messaging [1] - Asynchronous messaging, no heartbeat [0] - Reserved. Must be 0b. The returned value is 0x6 (Synchronous and Asynchronous support without heartbeat).
uint8	numberEventClassReturned The count of <i>eventClass</i> enumerated bytes returned in this response. Returns 5 - supported events listed below.
uint8	eventClass [0] 0x00 = sensorEvent
uint8	eventClass [1] 0x02 = redfishTaskExecuteEvent
uint8	eventClass [2] 0x03 = redfishMessageEvent
uint8	eventClass [3] 0x04 = pldmPDRRepositoryChgEven
uint8	eventClass [4] 0x05 = pldmMessagePollEvent

10.8.2.1.7 EventMessageBufferSize (0xD)

Table 10-46. EventMessageBufferSize Command and Response

Type	Request Data
uint16	eventReceiverMaxBufferSize This is the maximum buffer to hold an event message transferred from the terminus to the event receiver. Minimum value allowed is 14 bytes.
Type	Response Data
enum8	completionCode Value: {PLDM_BASE_CODES}
uint16	terminusMaxBufferSize Return 2048 (2K buffers).

The size used to report events should be min (*eventReceiverMaxBufferSize*, *terminusMaxBufferSize*)
This command is medium dependent and different values can be received over PCIe and SMBus.

An INVALID_DATA error is reported if *eventReceiverMaxBufferSize* value is smaller than 14 bytes.

10.8.2.1.8 SetNumericSensorEnable (0x10)

The SetNumericSensorEnable command is supported for the Temperature & Link Speed sensors and is used to enable or disable generation of events from these sensors. The command and response are described in [Table 10-47](#).

The default at init time should be to have all sensors enabled.

Table 10-47. SetNumericSensorEnable Command and Response

Type	Request Data
uint16	sensorID Only accepted values are the sensor IDs of Thermal sensors and Link Speed sensors. Otherwise, an INVALID_SENSOR_ID error code is returned.
enum8	sensorOperationalState The desired state of the sensor value: {enabled, disabled, unavailable} Defines if the sensor is enabled, disabled or unavailable. Default is enabled.
enum8	sensorEventMessageEnable This value is used to enable or disable event message generation from the sensor. Possible values: { noChange, disableEvents, enableEvents, enableOpEventsOnly, enableStateEventsOnly } The relevant values are: 0x0 = noChange 0x1 = disableEvents 0x2 = enableEvents 0x3 = enableOpEventsOnly For other values, return an EVENT_GENERATION_NOT_SUPPORTED error.

Table 10-47. SetNumericSensorEnable Command and Response [continued]

Type	Response Data
enum8	completionCode Value: <pre> { PLDM_BASE_CODES, INVALID_SENSOR_ID = 0x80, INVALID_SENSOR_OPERATIONAL_STATE = 0x81, EVENT_GENERATION_NOT_SUPPORTED = 0x82 // an attempt was made to enable or disable event generation for a sensor that does not support event message generation }</pre>

10.8.2.1.9 GetSensorReading (0x11)

The GetSensorReading command is supported for the Temperature, Link speed, and power consumption sensors, and is used to read the current value of the sensor.

The default at init time should be to have all sensors enabled.

Note: Firmware updates sensor states and responds immediately when processing a GetSensorReading command. Hence, in case this command is received before next firmware polling and sensor states are updated, PlatformEventMessage is sent for this particular event.

In rare cases in which old events are still in the internal queue when the GetSensorReading command is received (i.e., retransmission due to transmit failure), BMC will get the stalled events.

The command and response are described in [Table 10-48](#).

Table 10-48. GetSensorReading Command and Response

Type	Request Data
uint16	sensorID Only accepted values are the IDs of Temperature, Link Speed, and Power Consumption sensors. Otherwise, an INVALID_SENSOR_ID error code is returned.
bool8	rearmEventState Ignored. All sensors are auto-rearm.
Type	Response Data
enum8	completionCode Possible values: <pre> { PLDM_BASE_CODES, INVALID_SENSOR_ID = 0x80, EARM_UNAVAILABLE_IN_PRESENT_STATE = 0x81 }</pre>
enum8	sensorDataSize The bit width and format of reading and threshold values that the sensor returns UINT8 (0x0) for thermal sensor and uint32 (0x4) for link speed.

Table 10-48. GetSensorReading Command and Response [continued]

enum8	sensorOperationalState The state of the sensor itself Possible values: <pre>{ enabled, disabled, unavailable, statusUnknown, failed, initializing, shuttingDown, inTest }</pre> Relevant values are: enabled Enabled and operating. The sensor is able to return valid <i>presentState</i>, <i>previousState</i>, <i>presentReading</i>, and <i>eventState</i> values. This state can be set through the SetNumericSensorEnable command. disabled The sensor is disabled from returning <i>presentReading</i> and event state values. This state is settable through the SetNumericSensorEnable command. unavailable The sensor should be ignored due to the configuration of the platform or monitored entity. For example, the sensor is for monitoring a processor temperature, but the processor is not installed. This state is settable through the SetNumericSensorEnable command. failed The sensor has failed. The sensor implementation has determined that it can not return correct values for one or more of its <i>presentState</i> or <i>eventState</i> values.
enum8	sensorEventMessageEnable Possible values: <pre>{ noEventGeneration, eventsDisabled, eventsEnabled, opEventsOnlyEnabled, stateEventsOnlyEnabled }</pre> as set in the SetNumericSensorEnable command
enum8	presentState The most recently assessed state value monitored by the sensor. If the <i>sensorOperationalState</i> is set to enabled the sensor must return a value other than "Unknown" for the <i>presentState</i>. If the <i>sensorOperationalState</i> is not set to enabled, the sensor returns "Unknown" for the <i>presentState</i>. See Section 10.8.3.2 for the reported state per sensor.
enum8	previousState The state that the <i>presentState</i> was entered from. See Section 10.8.3.2 for the reported state per sensor. If the <i>sensorOperationalState</i> is not set to enabled, the sensor returns "Unknown" for the <i>previousState</i>.
enum8	eventState Same as <i>presentState</i> for all sensors but the thermal sensor (assume no hysteresis). For Thermal sensor return the value of the last crossed threshold including thresholds considerations as described in Section 17.9 of DSP0248.
uint8 uint32	presentReading The present value indicated by the sensor See Section 10.8.3.2 for the reported value per sensor.

10.8.2.1.10 GetSensorThresholds (0x12)

The GetSensorThresholds command is supported for the Temperature sensors and is used to read the current values of the sensor thresholds. The command and response are described in [Table 10-49](#).

Table 10-49. GetSensorThresholds Command and Response

Type	Request Data
uint16	sensorID Only accepted values are the Temperature Sensor IDs. Otherwise, an INVALID_SENSOR_ID error code is returned.
Type	Response Data
enum8	completionCode Possible values: { PLDM_BASE_CODES, INVALID_SENSOR_ID = 0x80 }
enum8	sensorDataSize uint8 (0x0)
uint8	upperThresholdWarning According to the value set in the SetSensorThresholds command. Default is according to the Normal Value.Reported Thermal Sensor <i>WarningHigh</i> field in NVM if Thermal Sensor Thresholds. <i>upperWarning</i> supported is set, zero otherwise.
uint8	upperThresholdCritical According to the value set in the SetSensorThresholds command. Default is according to the Warning Value.Reported Thermal Sensor <i>CriticalHigh</i> field in NVM if Thermal Sensor Thresholds. <i>upperCritical</i> supported is set, zero otherwise.
uint8	upperThresholdFatal According to the value set in the SetSensorThresholds command. Default is according to the Critical Value.Reported Thermal Sensor <i>FatalHigh</i> field in NVM if Thermal Sensor Thresholds. <i>upperFatal</i> supported is set, zero otherwise.
uint8	lowerThresholdWarning Value: 0
uint8	lowerThresholdCritical Value: 0
uint8	lowerThresholdFatal Value: 0

10.8.2.1.11 SetSensorThresholds (0x13)

The SetSensorThresholds command is supported for the Temperature sensors and is used to set the values of the sensor thresholds. The command and response are described in [Table 10-50](#).

Table 10-50. SetSensorThresholds Command and Response

Type	Request Data
uint16	sensorID Only accepted values are the Temperature Sensor IDs. Otherwise, an INVALID_SENSOR_ID error code is returned.
enum8	sensorDataSize If the value is different than uint8 (0x0), return an ERROR_INVALID_DATA error code is returned
For sensorDataSize = uint8 or sint8	

Table 10-50. SetSensorThresholds Command and Response [continued]

uint8 sint8	upperThresholdWarning
uint8 sint8	upperThresholdCritical
uint8 sint8	upperThresholdFatal
uint8 sint8	lowerThresholdWarning Ignored.
uint8 sint8	lowerThresholdCritical Ignored.
uint8 sint8	lowerThresholdFatal Ignored.
Type	Response Data
enum8	completionCode Possible values: { PLDM_BASE_CODES, INVALID_SENSOR_ID = 0x80 }

10.8.2.1.12 GetSensorHysteresis (0x15)

The GetSensorHysteresis command is supported for the Temperature sensors and is used to read the values of the sensor hysteresis. For other sensors, an INVALID_SENSOR_ID is returned. The command and response are described in [Table 10-51](#).

Table 10-51. GetSensorHysteresis Command and Response

Type	Request Data
uint16	sensorID Accepted values are the Temperature Sensor IDs. Otherwise, an INVALID_SENSOR_ID error code is returned.
Type	Response Data
enum8	completionCode Possible values: { PLDM_BASE_CODES INVALID_SENSOR_ID = 0x80 }
enum8	sensorDataSize The bit width of the hysteresis value that is being returned. Value: 0 (uint8)
uint8 sint8	hysteresisValue Read from <i>Hysteresis</i> field in Thermal Sensor Tolerance & Hysteresis NVM word for the relevant sensor.

10.8.2.1.13 SetStateSensorEnables (0x20)

The SetStateSensorEnables command is supported for the link state, health state, presence, configuration change, thermal Trip, and power consumption sensors and is used to enable or disable generation of events from these sensors. The command and response are described in [Table 10-52](#).

Table 10-52. SetStateSensorEnables Command and Response

Type	Request Data
uint16	sensorID Accepted values the sensor IDs of the supported sensors described above. Otherwise, an INVALID_SENSOR_ID error code is returned.
uint8	compositeSensorCount Number of <i>OpFields</i> present. If the value is higher than the number of sensors within the composite sensor, an ERROR_INVALID_DATA error code is returned. If the value is lower than the number of sensors within the composite sensor, return info for the first <i>compositeSensorCount</i> sensors The accessed sensors are 0- <i>compositeSensorCount</i> -1
<i>compositeSensorCount</i> instances of <i>OpFields</i> :	
enum8	sensorOperationalState The desired state of the sensor. Defines if the sensor is enabled, disabled or unavailable. Possible values: <pre>{ enabled, disabled, unavailable }</pre> Default is enabled.
enum8	EventMessageEnable This value is used to enable or disable event message generation from the sensor. Possible values: <pre>{ noChange, disableEvents, enableEvents, enableOpEventsOnly, enableStateEventsOnly }</pre> The relevant values are: 0x0 = noChange 0x1 = disableEvents 0x2 = enableEvents 0x4 = enableStateEventsOnly — For health state, thermal trip, configuration, configuration changed, present and link state sensors 0x0 = noChange — For power sensor. For other values, return an EVENT_GENERATION_NOT_SUPPORTED error. <i>enableEvents</i> and <i>enableStateEventsOnly</i> are treated the same.
Type	Response Data
enum8	completionCode Possible values: <pre>{ PLDM_BASE_CODES INVALID_SENSOR_ID = 0x80, EVENT_GENERATION_NOT_SUPPORTED = 0x82 }</pre>

10.8.2.1.14 GetStateSensorReadings (0x21)

The GetStateSensorReadings command is supported for all the state sensors (Composite network controller sensor, Composite NIC sensor, Port link sensor, Cable Presence Sensor, Plug Composite Sensor) and is used to read the current state of the sensors. The command and response are described in [Table 10-53](#).

Table 10-53. GetStateSensorReadings Command and Response

Type	Request Data
uint16	sensorID Accepted values are all state sensors IDs. Otherwise, an INVALID_SENSOR_ID error code is returned.
bitfield8	sensorRearm Ignored. All sensors are auto-rearm.
uint8	Reserved Value = 0x00
Type	Response Data
enum8	completionCode Possible values: { PLDM_BASE_CODES, INVALID_SENSOR_ID = 0x80 }
uint8	compositeSensorCount Return the number of sensors within the composite sensor.
Repeated once per sensor within the composite sensor.	
enum8	sensorOperationalState The state of the sensor itself as set using the SetStateSensorEnables command.
enum8	presentState This field is used to return a state value from a PLDM State Set that is associated with the sensor. The value reflects the most recently assessed state. See Section 10.8.3.2 for the state events reported.
enum8	previousState The state that the <i>presentState</i> was entered from. See Section 10.8.3.2 for the reported state per sensor. If the <i>sensorOperationalState</i> is not set to enabled, the sensor returns “Unknown” for the <i>previousState</i> .
enum8	eventState same as <i>presentState</i> .

10.8.2.1.15 GetPDRRepositoryInfo (0x50)

The GetPDRRepositoryInfo is used to provide a generic description of the PDRs in the device. The command and response are described in [Table 10-54](#).

Table 10-54. GetPDRRepositoryInfo Command and Response

Type	Request Data
---	None.
Type	Response Data
enum8	completionCode Value: {PLDM_BASE_CODES}
enum8	repositoryState Possible values: { available, // Record data can be read from the repository. updateInProgress, // Record data is unavailable because an update is in progress. failed // Record data is unavailable because of a detected failure condition. } Should be available (0x0) in regular cases. If a configuration change was detected (cable plug, link configuration change), might return <i>updateInProgress</i> (0x1).
timestamp104	updateTime This time stamp identifies when the standard PDR Repository data was originally created, or the time of the most recent update if the data has been updated after it was created. This time does not include changes of PDRs that have a PDR Type of "OEM". Returns the time in internal clock of the firmware of the last PDR update.
timestamp104	OEMUpdateTime This time stamp identifies when OEM PDRs in the PDR Repository were originally created, or the time of the most recent update if the data has been updated after it was created. Currently no OEM PDRs are defined, so return 0.
uint32	recordCount Total number of PDRs in this repository.
uint32	repositorySize Size of the PDR Repository in bytes. This value provides information that can be used for helping estimate buffer size requirements when accessing PDRs. This size covers only the cumulative sizes of the PDR record fields. This size does not include the size for any internal header structures that are used for maintaining the PDRs. This number does not report and may not directly correlate to the amount of internal storage used for PDRs because. For example, an implementation might elect to internally compress or use other encodings of the PDR data. An implementation is allowed to round this number up to the nearest kilobyte (1024 bytes). Return the size of the entire PDR data structure rounded up to 1K.
uint32	largestRecordSize Size of the largest record in the PDR Repository in bytes. This value provides information that can be used for helping estimate buffer size requirements when accessing PDRs. An implementation is allowed to round this number of up to the nearest 64-byte increment. Return largest PDR size in bytes rounded up to 128 bytes.

Table 10-54. GetPDRRepositoryInfo Command and Response [continued]

uint8	dataTransferHandleTimeout The minimum interval, in seconds, that a <i>dataTransferHandle</i> value remains valid after it was delivered in the response of a GetPDR or FindPDR command. Special values: 0x00 = No timeout 0x01 = Default minimum timeout 0xFF = Timeout >254 seconds. Any timeout values that are less than the specified default minimum timeout are illegal. Return 0x01 (default timeout of 30 seconds).

10.8.2.1.16 GetPDR (0x51)

The GetPDRRepositoryInfo is used to access the PDRs in the device. The command and response are described in [Table 10-55](#).

The PDR data structures used to fill the *recordData* in this command are described in [Section 10.8.3.3](#).

Table 10-55. GetPDR Command and Response

Type	Request Data
uint32	recordHandle See Section 10.8.3.3 for details of the PDR to return for each handle. A value of 0x0000_0000 returns the NIC association PDR (1100).
uint32	dataTransferHandle Reflects the offset in bytes of the data within the PDR. Special value: {use 0x0000_0000 if the <i>transferOperationFlag</i> is <i>GetFirstPart</i> }
enum8	transferOperationFlag Indicates whether this request is for the first portion of the PDR. Possible values: { GetNextPart = 0x00, GetFirstPart = 0x01 }
uint16	requestCount The maximum number of record bytes requested to be returned in the response to this instance of the GetPDR command. Note: The responder may return fewer bytes than were requested.
uint16	recordChangeNumber If the <i>transferOperationFlag</i> field is set to <i>GetFirstPart</i> , set this value to 0x0000. If the <i>transferOperationFlag</i> field is set to <i>GetNextPart</i> , set this to the <i>recordChangeNumber</i> value that was returned in the header data from the first part of the PDR
Type	Response Data
enum8	completionCode Possible values: { PLDM_BASE_CODES, INVALID_DATA_TRANSFER_HANDLE = 0x80, INVALID_TRANSFER_OPERATION_FLAG = 0x81, INVALID_RECORD_HANDLE = 0x82, INVALID_RECORD_CHANGE_NUMBER = 0x83, TRANSFER_TIMEOUT = 0x84, REPOSITORY_UPDATE_IN_PROGRESS = 0x85 }

Table 10-55. GetPDR Command and Response [continued]

uint32	nextRecordHandle The next record handle is based on Table 10-63 , where the handles are returned in ascending values. Note: When RDE is enabled, an additional list of PRDs is added, as described in Section 10.8.3.3.7 . Special value: 0x0000_0000 = No more PDRs following this one.
uint32	nextDataTransferHandle A handle that identifies the next portion of the PDR data to be transferred, if any portions are remaining. Reflects the offset in bytes of the next data within the PDR Special value: Returns 0x0000_0000 if there is no remaining data.
enum8	transferFlag Indicates what portion of the PDR is being transferred Possible values: { Start = 0x00, Middle = 0x01, End = 0x04, StartAndEnd = 0x05 } Set according to the requestCount.value.
uint16	responseCount The number of <i>recordData</i> bytes returned in this response. Special value: Returns 0x0000 if the <i>requestCount</i> was 0x0000.
(var)	recordData PDR data bytes. This field is absent if <i>responseCount</i> =0x0000. The number of PDR data bytes returned in this field must match <i>responseCount</i> .
If <i>transferFlag</i> = End	
uint8	ransferCRC A CRC-8 for the overall PDR. This is provided to help verify data integrity for a PDR when it is transferred using a multi-part transfer. The CRC is calculated over the entire PDR data using the polynomial $x^8 + x^2 + x^1 + 1$ (This is the same polynomial used in the MCTP over SMBus/I ² C transport binding specification). The CRC is calculated from most-significant bit to least-significant bit on bytes in the order that they are received. This field is only present when <i>transferFlag</i> =End.

10.8.2.1.17 GetPDRRepositorySignature (0x53)

The PDR Repository Signature is a value that represents the entire collection of terminus Platform Device Records (PDRs). This is different than the GetPDRRepositoryInfo command because only an opaque 32-bit value is returned. The purpose of the PDR Repository Signature is to provide the management controller the capability to determine if a terminus PDR repository has changed during state transitions such as power cycles.

Table 10-56. GetPDRRepositorySignature Command and Response

Type	Request Data
-	-
Type	Response Data
enum8	completionCode Value: {PLDM_BASE_CODES}
uint32	pdrRepositorySignature This is a 32-bit value and remains persistent unless a change is detected in any record of the PDR repository. This field should be computed once (during device boot) as low 32 bits of SHA256 of PDRs

10.8.2.2 PLDM Monitoring and Control Events

The following events can be generated by the E610:

- **sensorEvent** — See DSP0248 for more details. This event is sent to BMC upon a change in the state of one of the sensors.
- **redfishTaskExecuteEvent** — See DSP0248 for more details. This event is sent to BMC upon long running task completion.
- **redfishMessageEvent** — See DSP0248 for more details. This event is sent to BMC in case a Redfish message should be sent.
- **pldmPDRRepositoryChgEvent** — This message is sent in case of a change in the PDR structure.
This event reports PDR change in runtime. This Event is to signal the PLDM Event Receiver that there is a change in the terminus PDR repository. The device returns the PDR Types or the PDR Record Handles for the PDRs to be retrieved from the terminus. This allows a simple method for a terminus to indicate which portion of its “virtual” PDR Repository needs to be refreshed. The PLDM terminus client (or event receiver) will need to comprehend additions, deletions and modifications of the PDRs as it updates the system primary PDR repository. See [Table 10-57](#) and [Table 10-58](#) for the event format.
- **pldmMessagePollEvent** — This event is returned in asynchronous mode only when the reported event cannot fit into a single PLDM message.

[Table 10-57](#) through [Table 10-62](#) describe the various events formats:

Table 10-57. PDR Repository Change Event Data

Type	Request Data
enum8	eventDataFormat Possible values: <pre>{ refreshEntireRepository, formatIsPDRTypes, formatIsPDRHandles }</pre> This field indicates if the <i>changedRecords</i> are of "PDR Types" or "PDR Record Handles". The device signals to the event receiver to re-enumerate the entire device PDR repository by supplying the value <i>refreshEntireRepository</i>. To signal that only certain types of PDRs should be refreshed, the device supplies the value <i>formatIsPDRTypes</i> and provides one change record below for each type of PDR to be refreshed.
uint8	numberOfChangeRecords The number of <i>changeRecords</i> (N_R) following this field. If the <i>eventDataFormat</i> is <i>refreshEntireRepository</i> , this value is zero.
var	changeRecord [0] See Table 10-58 for details. This field is not present if the <i>numberOfChangeRecords</i> is zero (0).
var	changeRecord [1]
...	...
var	changeRecord [$N_R - 1$]

Table 10-58. pldmPDRRepositoryChgEvent changeRecord Format

Type	Request Data
enum8	eventDataOperation Possible values: <pre>{ refreshAllRecords, recordsDeleted, recordsAdded, recordsModified }</pre> For each <i>pldmPDRRepositoryChgEvent</i> record, there can only be a single operation. This simplifies the parsing for both the terminus and the event receiver. The order the event records are provided is "RefreshAll", "Deleted", "Added", "Modified". The value <i>refreshAllRecords</i> is only supplied when <i>eventDataFormat</i> was set to <i>formatIsPDRTypes</i>. In this case, the entries below represent a series of PDR types to be refreshed.
uint8	numberOfChangeEntries The number of change entries (N_E) following this field.
uint32	changeEntry [0] This value is either a "PDR Type" enumeration or a "PDR Record Handle" as counted by the <i>eventDataFormat</i> field in the <i>pldmPDRRepositoryChgEvent</i> event message. There can be multiple PDR Types (such as Numeric Sensor, State Sensor, and Entity Association Sensor) to be retrieved due to a "hot plug" event for the terminus. All the changed PDR Types can be returned in a single event message. The client (or event receiver) can use the FindPDR command to gather the PDR record. Alternatively, the terminus can provide a list of PDR Record Handles, which the MC can use as input to the GetPDR command.
uint32	changeEntry [1]
...	...
uint32	changeEntry [$N_E - 1$]

Table 10-59. redfishTaskExecutedEvent Class eventData Format

Type	Request Data
uint32	resourceID ResourceID associated with the Task that has completed execution as received in RDeOperationInit RDE command.
uint16	operationID Operation associated with the Task that has completed execution as received in RDeOperationInit RDE command.

Table 10-60. redfishMessageEvent Class eventData Format

Type	Request Data
uint8	eventCount The number of Redfish Events (N) encoded in the <i>eventData</i> field below.
uint16	eventDataLength Length in bytes of the <i>eventData</i> field below, which comprises the encoding of one or more Redfish Events contained within this PLDM event. This value must not exceed the negotiated event message size.
uint32	resourceID [0] An opaque handle referencing the particular collection of schema-based Redfish data associated with the first Redfish Event encoded in the <i>eventData</i> field below.
enum8	eventSeverity [0] The severity of the first Redfish Event in the Redfish EventRecords array encoded in <i>eventData</i> below. Values: 0 = OK 1 = Warning 2 = Critical
uint32	resourceID [N - 1] An opaque handle referencing the particular collection of schema-based Redfish data associated with the last Redfish Event encoded in the <i>eventData</i> field below.
enum8	eventSeverity [N - 1] The severity of the last Redfish Event in the Redfish EventRecords array encoded in <i>eventData</i> below. Values: 0 = OK 1 = Warning 2 = Critical
bejEncoding	eventData BEJ encoded Event payload data. The <i>bejEncoding</i> PLDM type is defined in DSP0218.

Table 10-61. pldmMessagePollEvent

Type	Request Data
uint8	formatVersion Version of the event format (the format and definition of the following bytes): 0x01 for this specification.
uint16	eventID Identifier for the event that requires multi-part transfer.
uint32	dataTransferHandle A handle that is used to identify the event data to be received via the PollForPlatformEventMessage command.

Table 10-62. redfishMessageEvent

Type	Request Data
uint8	eventCount The number of Redfish Events N encoded in the <i>eventData</i> field below.
uint16	eventDataLength Length in bytes of the <i>eventData</i> field below, which comprises the encoding of one or more Redfish Events contained within this PLDM event. This value must not exceed the negotiated event message size.
uint32	resourceID [0] An opaque handle referencing the particular collection of schema-based Redfish data associated with the first Redfish Event encoded in the <i>eventData</i> field below.
enum8	eventSeverity[0] The <i>eventSeverity</i> allows a client to indicate severity to logs or other mechanism. Values: 0 = OK 1 = Warning 2 = Critical Each Registry Message contains severity field.
...	...
uint32	resourceID [N - 1] An opaque handle referencing the particular collection of schema-based Redfish data associated with the last Redfish Event encoded in the <i>eventData</i> field below.
enum8	eventSeverity [N - 1] The severity of the last Redfish Event in the Redfish EventRecords array encoded in <i>eventData</i> below. Values: 0 = OK 1 = Warning 2 = Critical Each Registry Message contains severity field.
bejEncoding	eventData BEJ encoded Event payload data.

The internal BEJ Event payload data is based on the Event schema.

The OriginOfCondition field in the Redfish event schema contains a link reference to a Redfish resource associated with a Redfish event.

10.8.2.3 PDR Dynamic Changes Flow

In the DSP2054 PLDM model, PDRs can change dynamically due to changes in the NIC configuration. The following events might cause a change in the PDR structure:

QSFP/SFP module plug/unplug events might cause a change in the PDR content.

In addition, transition from breakout to non-breakout cable configuration and vice versa changes the PRD structure, but it is not a dynamic flow. See section [Section 10.8.2.3.1](#) on the events that indicate these events.

Once a topology change impacting the PDR structure is detected, the following logical flow should be applied:

1. For each record that changed, increase the *recordChangeNumber*.
2. Update the *updateTime* used in the *GetPDRRepositoryInfo* answer to reflect the current time (using the free running timer of the firmware).

3. Change the state of the Configuration Change sensor in the NIC composite sensor to Configuration Change Detected.

10.8.2.3.1 PDR Change Events

10.8.2.3.1.1 QSFP/SFP Plug Events

A module plug event creates a “Media Inserted” event within the firmware. The parts of the PDR that can be impacted by this event are: Plug Thermal Sensor, Power Consumption, and Port Speed. These must be populated with values relevant to the module.

If the inserted module does not support thermal reading, the thermal sensor behaves as if no module is inserted (see [Section 10.8.2.3.1.2](#)).

10.8.2.3.1.2 QSFP/SFP unplug events

A module unplug event creates a “Media Removed” event within the firmware.

When a pluggable module is not present, a query of the module’s numeric sensors (power and temperature) via *GetSensorReading* is replied with *sensorOperationalState* value of Unavailable. A query of the module’s Health State or Thermal Trip State sensors via *GetStateSensorReading* is replied with *sensorOperationalState* value of Unavailable, and *presentState* and *previousState* values of Unknown. The Presence State sensor *presentState* returns 0x2 (not-present).

The following PDRs values are updated upon module removal:

- Plug Thermal Sensor PDR:
 - All values in PDR from *unitModifier* onwards must be 0.
- Pluggable Module Power Consumption PDR:
 - *Maxreadable* must be 0.
 - *NominalValue* must be 0.
- Port (link) Speed PDR:
 - *Maxreadable* must be 0.
 - *NominalValue* must be 0.

10.8.2.3.1.3 Transition Between Breakout to Non-Breakout Cable Configurations

These transitions are initiated by the user after a media-conflict event notification. In this case, a reboot of the system is required, causing a PERST# event that in turn causes a re-read of the topology and a “Port Mode Change” event. Once a “Port Mode Change” event is received, the Link Association PDR should be updated accordingly.

10.8.3 PLDM Monitoring and Control - Generic Structures

10.8.3.1 Sensors Numbering

DSP2054 defines a set of IDs for each type of sensor, as in [Table 10-63](#):

Table 10-63. Sensors and Handles Numbering

Item	Max Count	Supported Number	Base Container ID	Max Container ID	Base Handle	Max Handle	Base Sensor ID	Max Sensor ID	Base Instance	Type ID
NIC	1	1	100		1100				1	68
Card Composite State Sensor	1	1			1101	1101	5	5	1	68
Connectors	8	4	$1040 + i*8$	$1047 + i*8$	$1110 + i*8$	$1117 + i*8$			$1 + i*8$	185
NIC Temp Sensors	10	3			$1130 + i*5$	$1139 + i*5$	$20 + i*5$	$29 + i*5$	$1 + i*5$	68
Network Controllers	$1 + i$	1	$1000 + i$	$1000 + i$	$1150 + i$	$1150 + i$			$1 + i$	144
Network Controller State	1	1			$1170 + i$	$1170 + i$	$60 + i$	$60 + i$	1	144
Ports of Network Controller	8	8			$1200 + i*8$	$1207 + i*8$			1	300
Link Speed of Network Controller	10	8			$1300 + i*8$	$1307 + i*8$	$100 + i*8$	$107 + i*8$	1	300
Link State of Network Controller	10	8			$1400 + i*8$	$1407 + i*8$	$200 + i*8$	$207 + i*8$	1	300
Temp Sensor per Network Controller	10	3			$1500 + i*10$	$1509 + i*10$	$300 + i*10$	$309 + i*10$	1	144
Plugs	8	8	$1010 + i*8$	$1017 + i*8$	$1600 + i*8$	$1607 + i*8$			1	206/ 207/ 211/ 212
Plug Power Sensor	8	8			$1700 + i*8$	$1707 + i*8$	$400 + i*8$	$407 + i*8$	1	206/ 207/ 211/ 212
Plug Temp Sensor	10	4			$1800 + i*8$	$1807 + i*8$	$500 + i*8$	$507 + i*8$	1	206/ 207/ 211/ 212
Plug Composite Sensor	8	8 (1 per plug)			$2000 + i*8$	$2007 + i*8$	$700 + i*8$	$707 + i*8$	1	206/ 207/ 211/ 212
Cable	8								1	187
Comm. Channel	8	8	$1060 + i*8$	$1067 + i*8$	$2100 + i*8$	$2107 + i*8$			$1 + i*8$	6

Calculated
Model Constant
User Parameter

Where:

Item	The Name of the Referenced Items
i	The offset value as read from the SDP indicated by the Instance ID SDP NVM field. If Instance ID from SDP is not set, i = 0.
Max Count	Max number of elements of this type allowed by the PLDM model.
Supported Number	The largest number of instances expected by this feature.
Base Container ID / Max Container ID	The first and last ID of the device as referenced in PDRs Base Handle.
Max Handle	The first and last ID of the PDRs describing the instances.
Base Sensor ID / Max Sensor-ID:	The first and last sensor ID of this sensor type
Base Instance	The first instance of this type as listed in the containing PDR.

These are **maximum** values. Device with fewer than eight ports, for example, have less than maximum value in many fields.

This maps to the following PDR/Sensor diagram shown in [Figure 10-12](#).

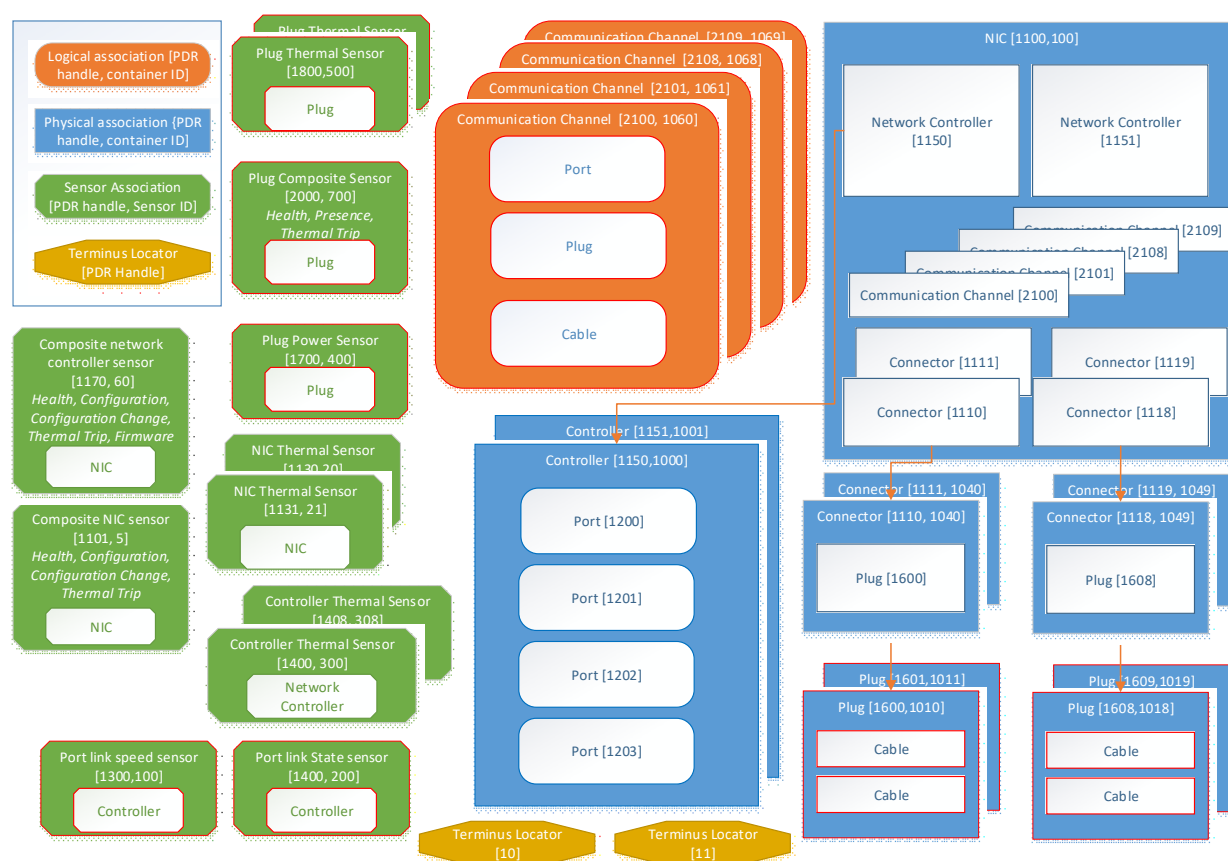


Figure 10-12. PDR/Sensor Diagram

Notes:

- The logic behind Configuration State and Configuration Changed is based on HP DCi input, so if a generic version is exposed, it should be based on other parameters.
- The PDRs with a red border are dynamic and may change due to module plug events or breakout cables insertion events.

10.8.3.2 Sensors

The following sensors are maintained in the device to be reported via the PLDM commands described in Section 10.8.2.1.

- Numeric Sensors:
 - Temperature Sensors
 - NIC/Controller/Plug
 - Link Speed Sensors
 - Plug power consumption Sensors

- Composite State Sensors:
 - NIC: {Health, Configuration, Configuration Change, Thermal Trip}
 - Controller: {Health, Configuration, Configuration Change, Thermal Trip}
 - Plug: {Health, Presence, Thermal Trip}
- Regular State Sensors:
 - Link state

The numbering of the sensors is according to [Section 10.8.3.1](#).

10.8.3.2.1 Numeric Sensors

10.8.3.2.1.1 Temperature Sensor Data Structure

There are three types of possible thermal sensors:

- Internal (on-die) thermal sensors
- External (on-board) thermal sensors
- Plug (SFP/QSFP) thermal sensors

The identification of the type is set according to the following Topology Netlist fields:

- Sensor Location at word 9 (Port Affinity High) of the Node Thermal Configuration Section of the Temperature Sensor node.
- Node Part Number at word 3 (Part Number and Node Options) of the Node Header Section of the Temperature Sensor node.

Sensor Type	Sensor Location	Node Part Number
Internal	0x01	0x36 = Embedded in Internal PHY
External (PHY)	0x02 - 0x07	Embedded in external PHY, e.g. 0x31 = C827 0x32 = X557 0x33 = 88E1543 0x34 = 88E1512 0x35 = 88E1514
Plug SFP (SFF-8472 compliant)	0x10	0x44 = Embedded in SFP plug
Plug QSFP (SFF-8636/8436 compliant)	0x11	0x45 = Embedded in QSFP plug

Notes:

- Firmware uses of Sensor Location to determine Sensor Type, Part Numbers is informative only, as the list may dynamically grow further.
- Temperature Sensors should be implemented only if the Expose Thermal Sensors bit in the PLDM control word is set. The number of thermal sensors exposed from each type is derived from the network topology. The current temperature for each sensor can be read through the GET_TEMP DNL script.

Type	Request Data
uint16	sensorID = According to Section 10.8.3.1 (20 + n, 300 + n, 500 for plug, NIC, Controllers sensors respectively).
enum8	sensorEventClass = numericSensorState (0x2) or sensorOpState (0x0) If the Sensor has a <i>sensorEventMessageEnable</i> value of <i>enableOpEventsOnly</i> , return a <i>sensorOpState</i> . If the Sensor has a <i>sensorEventMessageEnable</i> value of <i>enableEvents</i> , return a <i>numericSensorState</i> . Otherwise, events are not generated.
enum8	presentState The <i>eventState</i> value from the state change that triggered the event message. If temperature is >= Fatal Threshold, Return <i>UpperFatal</i> (0xA) If temperature is >= Critical Threshold return <i>UpperCritical</i> (0x9). If temperature is >= Warning Threshold return <i>UpperWarning</i> (0x8) Otherwise, return Normal (0x1). Thresholds are set by the SetSensorThresholds command.
enum8	previousState The <i>eventState</i> value for the state from which the present state was entered. Return the state that was reported in previous events. Special value: This value must be set to the same value as <i>presentState</i> if the <i>previousState</i> is unknown (which may be the case for events that are generated on the first status assessment that occurs after a sensor has been initialized).
enum8	sensorDataSize = 0x0 (uint8) - returned only if <i>sensorEventClass</i> = <i>numericSensorState</i> (0x2).
uint8	presentReading Temperature. The resolution is according to the resolution defined in the Thermal Sensor PDR (Section 10.8.3.3.6.1) - returned only if <i>sensorEventClass</i> = <i>numericSensorState</i> (0x2).

10.8.3.2.1.2 Link Speed Data Structure

Type	Request Data
uint16	sensorID = 99 + Port#
enum8	sensorEventClass = numericSensorState (0x2) or sensorOpState (0x0) If the Sensor has a <i>sensorEventMessageEnable</i> value of <i>enableOpEventsOnly</i> , return a <i>sensorOpState</i> . If the Sensor has a <i>sensorEventMessageEnable</i> value of <i>enableEvents</i> , return a <i>numericSensorState</i> . Otherwise, events are not generated.
enum8	presentState Return Normal (0x1).
enum8	previousState Return Normal (0x1).
enum8	sensorDataSize = uint32 (0x4) - returned only if <i>sensorEventClass</i> = <i>numericSensorState</i> (0x2).
uint8	presentReading Link Speed in Megabits per second - returned only if <i>sensorEventClass</i> = <i>numericSensorState</i> (0x2).

10.8.3.2.1.3 Plug Power Data Structure

Type	Request Data
uint16	sensorID = 399 + Plug#
enum8	sensorEventClass = numericSensorState (0x2) or sensorOpState (0x0) If the Sensor has a <i>sensorEventMessageEnable</i> value of <i>enableOpEventsOnly</i> , return a <i>sensorOpState</i> . If the Sensor has a <i>sensorEventMessageEnable</i> value of <i>enableEvents</i> , return a <i>numericSensorState</i> . Otherwise, events are not generated.
enum8	presentState Return Normal (0x1).
enum8	previousState Return Normal (0x1).
enum8	sensorDataSize = uint32 (0x4) - returned only if <i>sensorEventClass</i> = <i>numericSensorState</i> (0x2).
uint8	presentReading Plug nominal power as read from addresses below in the module EEPROM through I ² C read. The reported value is translated to 0.1 Watts units. SFPx - SFF-8472 v12.2 - 0xA0 - byte 64 in Table 8-3. QSFPx - SFF-8636 v2.7 - 0xA0 - byte 129 in Table 6-16. Returned only if <i>sensorEventClass</i> = <i>numericSensorState</i> (0x2). The value is returned from the <i>sff_get_pwr</i> (activity ID = 0x0027) DNL script.

10.8.3.2.2 NIC Composite State Sensors

The NIC Composite State Sensors are exposed only if PLDM Offset is zero (Instance ID SDP is not present or set to 0) and Expose NIC Composite State Sensor bit is set in PLDM Control NVM word.

10.8.3.2.2.1 NIC Health State Data Structure

Type	Request Data
uint16	sensorID = 5
enum8	sensorEventClass = stateSensorState (0x1)
uint8	sensorOffset = 0x0 (offset in NIC composite state structure)
enum8	presentState Note: All thermal sensing events here relate to the all the registered sensors. The state reflects the worst state of all sensors. The <i>eventState</i> value from the state change that triggered the event message. Return 0x1 (Normal) if all NVM checks are OK and thermal sensor value is below the warning threshold. Return 0x5 - Upper Non-Critical, if the thermal sensor value is above the warning threshold and below the Critical Threshold. Return 0x7 - Upper Critical, if the thermal sensor value is above the critical threshold and below the Fatal Threshold. Return 0x9 - Upper Fatal, if the thermal sensor value is above the fatal threshold Return 0x0 (Unknown) otherwise. Note: Changes to the Health sensor are coordinated with thermal events. So events will occur according to the thermal sensor hysteresis rules.

10.8.3.2.2.2 NIC Configuration Data Structure

Type	Request Data
uint16	sensorID = 5
enum8	sensorEventClass = stateSensorState (0x1)
uint8	sensorOffset = 0x1 (offset in NIC composite state structure)
enum8	presentState The <i>eventState</i> value from the state change that triggered the event message. One of the following: 1 = Valid Configuration 2 = Invalid Configuration TBD - For now Invalid configuration means that the firmware failed to find all the needed TLVs in the PFA or other associated errors.
enum8	previousState The <i>eventState</i> value for the state from which the present state was entered. Use the same logic as for the present state, only for the previous state. Special value: This value must be set to the same value as <i>presentState</i> if the <i>previousState</i> is unknown (which may be the case for events that are generated on the first status assessment that occurs after a sensor has been initialized).

10.8.3.2.2.3 NIC Changed Configuration State Data Structure

Type	Request Data
uint16	sensorID = 5
enum8	sensorEventClass = stateSensorState (0x1)
uint8	sensorOffset = 0x2 (offset in NIC composite state structure)
enum8	presentState The <i>eventState</i> value from the state change that triggered the event message. One of the following: 1 = Normal. Initial State. 2 = Configuration Change Detected. If a change in the PDR structure was detected. Move back to Normal after State is read.
enum8	previousState The <i>eventState</i> value for the state from which the present state was entered. Use the same logic as for the present state, only for the previous state. Special value: This value must be set to the same value as <i>presentState</i> if the <i>previousState</i> is unknown (which may be the case for events that are generated on the first status assessment that occurs after a sensor has been initialized).

10.8.3.2.2.4 NIC Thermal Trip State Data Structure

Type	Request Data
uint16	sensorID = 5
enum8	sensorEventClass = stateSensorState (0x1)
uint8	sensorOffset = 0x3 (offset in NIC composite state structure)
enum8	presentState Note: All thermal sensing events here relate to the all the registered sensors. The state reflects the worst state of all sensors. The <i>eventState</i> value from the state change that triggered the event message. Return 0x1 (Normalalways). Note: If auto shutdown will be implemented, will also support Thermal trip state (0x2).

10.8.3.2.3 Controller Composite State Sensors

10.8.3.2.3.1 Controller Health State Data Structure

Type	Request Data
uint16	sensorID = 60 + PLDM_offset
enum8	sensorEventClass = stateSensorState (0x1)
uint8	sensorOffset = 0x0 (offset in Controller composite state structure)
enum8	presentState Note: All thermal sensing events here relate to the controller internal TS. The <i>eventState</i> value from the state change that triggered the event message. Return 0x1 (Normal) if all NVM checks are OK and thermal sensor value is below the warning threshold. Return 0x5 - Upper Non-Critical, if the thermal sensor value is above the warning threshold and below the Critical Threshold. Return 0x7 - Upper Critical, if the thermal sensor value is above the critical threshold and below the Fatal Threshold. Return 0x9 - Upper Fatal, if the thermal sensor value is above the fatal threshold. Return 0x0 (Unknown) otherwise. Note: Changes to the Health sensor are coordinated with thermal events. So events will occur according to the thermal sensor hysteresis rules.
enum8	previousState The <i>eventState</i> value for the state from which the present state was entered. Use the same logic as for the present state, only for the previous state. Special value: This value must be set to the same value as <i>presentState</i> if the <i>previousState</i> is unknown (which may be the case for events that are generated on the first status assessment that occurs after a sensor has been initialized).
enum8	previousState The <i>eventState</i> value for the state from which the present state was entered. Use the same logic as for the present state, only for the previous state. Special value: This value must be set to the same value as <i>presentState</i> if the <i>previousState</i> is unknown (which may be the case for events that are generated on the first status assessment that occurs after a sensor has been initialized).

10.8.3.2.3.2 Controller Configuration Data Structure

Type	Request Data
uint16	sensorID = 60 + PLDM_offset
enum8	sensorEventClass = stateSensorState (0x1)
uint8	sensorOffset = 0x1 (offset in Controller composite state structure)
enum8	presentState The <i>eventState</i> value from the state change that triggered the event message. One of the following: 1 = Valid Configuration 2 = Invalid Configuration Invalid configuration means that the firmware failed to find all the needed TLVs in the PFA or other associated errors.
enum8	previousState The <i>eventState</i> value for the state from which the present state was entered. Use the same logic as for the present state, only for the previous state. Special value: This value must be set to the same value as <i>presentState</i> if the <i>previousState</i> is unknown (which may be the case for events that are generated on the first status assessment that occurs after a sensor has been initialized).

10.8.3.2.3.3 Controller Configuration Change Data Structure

Type	Request Data
uint16	sensorID = 60 + PLDM_offset
enum8	sensorEventClass = stateSensorState (0x1)
uint8	sensorOffset = 0x2 (offset in Controller composite state structure)
enum8	presentState The <i>eventState</i> value from the state change that triggered the event message. One of the following: 1 = Normal. Initial State. 2 = Configuration Change Detected. Returned if a change in the one of the channels associations was detected, or an NVM update occurred. Move back to Normal after State is read. An NVM update is detected by a Shadow RAM dump command
enum8	previousState The <i>eventState</i> value for the state from which the present state was entered. Use the same logic as for the present state, only for the previous state. Special value: This value must be set to the same value as <i>presentState</i> if the <i>previousState</i> is unknown (which may be the case for events that are generated on the first status assessment that occurs after a sensor has been initialized).

10.8.3.2.3.4 Controller Thermal Trip State Data Structure

Type	Request Data
uint16	sensorID = 60 + PLDM_offset
enum8	sensorEventClass = stateSensorState (0x1)
uint8	sensorOffset = 0x3 (offset in Controller composite state structure)
enum8	presentState Note: All thermal sensing events here relate to the controller internal TS. The <i>eventState</i> value from the state change that triggered the event message. Return 0x1 (Normal).

10.8.3.2.3.5 Firmware Version State Data Structure

Type	Request Data
uint16	sensorID = 60 + PLDM_offset
enum8	sensorEventClass = stateSensorState (0x1)
uint8	sensorOffset = 0x4 (offset in Controller composite state structure)
enum8	presentState 1 = Normal. Returned if there is no pending firmware upgrade. 2 = Version change detected - Compatible. Returned if there is a new non-activated firmware pending.
enum8	previousState The <i>eventState</i> value for the state from which the present state was entered. Use the same logic as for the present state, only for the previous state. Special value: This value must be set to the same value as <i>presentState</i> if the <i>previousState</i> is unknown (which might be the case for events that are generated on the first status assessment that occurs after a sensor has been initialized).

10.8.3.2.4 Plug Composite State Sensors

10.8.3.2.4.1 Plug Health State Data Structure

Type	Request Data
uint16	sensorID = 699 + PLDM_Offset*8 + Plug# (1-8)
enum8	sensorEventClass = stateSensorState (0x1)
uint8	sensorOffset = 0x0 (offset in Plug composite state structure)
enum8	presentState The <i>eventState</i> value from the state change that triggered the event message. Return 0x1 (Normal) if thermal sensor value is below the warning threshold. Return 0x5 - Upper Non-Critical, if the thermal sensor value is above the warning threshold and below the Critical Threshold. Return 0x7 - Upper Critical, if the thermal sensor value is above the critical threshold and below the Fatal Threshold. Return 0x9 - Upper Fatal, if the thermal sensor value is above the fatal threshold Return 0x0 (Unknown) otherwise Note: Changes to the Health sensor are coordinated with thermal events. So events will occur according to the thermal sensor hysteresis rules. TBD how to fix - For now equivalent to the crossed threshold of the plug thermal sensor.
enum8	previousState The <i>eventState</i> value for the state from which the present state was entered. Use the same logic as for the present state, only for the previous state. Special value: This value must be set to the same value as <i>presentState</i> if the <i>previousState</i> is unknown (which may be the case for events that are generated on the first status assessment that occurs after a sensor has been initialized).

10.8.3.2.4.2 Plug Presence State Data Structure

Type	Request Data
uint16	sensorID = 699 + PLDM_Offset*8 + Plug# (1-8)
enum8	sensorEventClass = stateSensorState (0x1)
uint8	sensorOffset = 0x1 (offset in Plug composite state structure)
enum8	presentState The <i>eventState</i> value from the state change that triggered the event message. Return 0x1 (Present) if the plug is connected. Return 0x2 (Not Present) if the plug is not connected. TBD if there should be plug or cable presence.
enum8	previousState The <i>eventState</i> value for the state from which the present state was entered. Use the same logic as for the present state, only for the previous state. Special value: This value must be set to the same value as <i>presentState</i> if the <i>previousState</i> is unknown (which may be the case for events that are generated on the first status assessment that occurs after a sensor has been initialized).

10.8.3.2.4.3 Plug Thermal Trip State Data Structure

Type	Request Data
uint16	sensorID = 699 + PLDM_Offset*8 + Plug# (1-8)
enum8	sensorEventClass = stateSensorState (0x1)
uint8	sensorOffset = 0x2 (offset in Plug composite state structure)
enum8	presentState <i>Note:</i> All thermal sensing events here relate to the plug internal TS. The <i>eventState</i> value from the state change that triggered the event message. Return 0x1 (Normal).
enum8	previousState The <i>eventState</i> value for the state from which the present state was entered. Use the same logic as for the present state, only for the previous state. Special value: This value must be set to the same value as <i>presentState</i> if the <i>previousState</i> is unknown (which may be the case for events that are generated on the first status assessment that occurs after a sensor has been initialized).

10.8.3.2.5 Simple Sensors

10.8.3.2.5.1 Port Link Sensor

Type	Request Data
uint16	sensorID = 199 + port# + PLDM_Offset*8
enum8	sensorEventClass = stateSensorState (0x1)
uint8	sensorOffset = 0x0 (not part of a composite)
enum8	presentState The <i>eventState</i> value from the state change that triggered the event message. Return 0x1 (Connected) if a link is present. Return 0x2 (Not Connected) if a link is not present.
enum8	PreviousState The eventState value for the state from which the present state was entered Use the same logic as for the present state, only for the previous state.

10.8.3.3 PDRs

The PDRs are numbered according to [Section 10.8.3.1, “Sensors Numbering”](#). The link list used to create the Next Handle Record, which should be created based on the exposed PDRs.

10.8.3.3.1 Terminus Locator PDR

Type	PDR Fields
uint32	recordHandle = 0x0000_0010 + PLDM_Offset
uint8	PDRHeaderVersion = 0x01
uint8	PDRType = 0x1 (Terminus Locator PDR)
uint16	recordChangeNumber = 0x0 (no changes expected in Terminus Locator PDR)
uint16	dataLength = 0x9
uint16	PLDMTerminusHandle = 0x0001 + PLDM_Offset
enum8	Validity = 0x1 (valid)
uint8	TID = PLDM Terminus ID as received by SetTID command (default is 0).
uint16	containerID = 0x0 (system)
enum8	terminusLocatorType = 0x1 (MCTP_EID)
enum8	terminusLocatorValueSize = 0x1 (one uint8)
<i>terminusLocatorValue</i> for <i>terminusLocatorType</i> = MCTP_EID:	
uint8	EID = The MCTP EID that is assigned to the device for the relevant medium.

10.8.3.3.2 Physical Association PDRs

The PDRs are exposed in a high-level representation for the association PDRs.

10.8.3.3.2.1 NIC Association PDR

A single NIC association PDR exists. The number of connectors exposed is based on the specific card configuration and is derived from the link topology (OCP 3.0 supports 1 or 2 connectors, but other cards might support more). The example below shows two connectors.

NIC Entity Association PDR		
Container ID	100	
Container Entity		
Entity Type	68	Add-in Card
Entity Instance Number	1	
Container Entity Container ID	0	
Association Type	Physical to Physical containment	
Contained Entity - Network Controller		
Entity Type	144	Network Controller
Entity Instance Number	1 + PLDM_Offset	
Container Entity Container ID	100	
Contained Entity - Connector		
Entity Type	185	Connector
Entity Instance Number	1 + PLDM_Offset*8	
Container Entity Container ID	100	
Contained Entity - Connector		
Entity Type	185	Connector
Entity Instance Number	2 + PLDM_Offset*8	
Container Entity Container ID	100	

10.8.3.3.2.2 Controller Association PDR

The controller includes all its ports. There can be up to eight ports for the E610. The actual number is derived from the link topology. The example below assumes two ports.

Network Controller Association PDR		
Container ID	1000 + PLDM_Offset	
Container Entity		
Entity Type	144	Network Controller
Entity Instance Number	1 + PLDM_Offset	
Container Entity Container ID	100	NIC
Association Type	Physical to Physical containment	
Contained Entity - Connector		
Entity Type	6	Communication Port
Entity Instance Number	1	
Container Entity Container ID	1000 + PLDM_Offset	
Contained Entity - Connector		
Entity Type	6	Communication Port
Entity Instance Number	2	
Container Entity Container ID	1000 + PLDM_Offset	

10.8.3.3.2.3 Connector Association PDR

A connector association PDR exists per cage of the device, as reflected from the topology file.

This PDR always exists only if a module is plugged. In case of module plug-in or plug-out event, the presence state event changes its state.

Connector #n Entity Association PDR (n = 1... number of cages)		
Container ID	1039 + n + PLDM_Offset*8	
Container Entity		
Entity Type	185	Connector
Entity Instance Number	n + PLDM_Offset*8	
Container Entity Container ID	100	NIC
Association Type	Physical to Physical containment	
Contained Entity - Plug		
Entity Type	206 (SFP28 Module) 207 (SFP+ Module) 211 (QSFP28 Module) 212 (QSFP+ Module)	Plug
Entity Instance Number	1	
Container Entity Container ID	1039 + n + PLDM_Offset*8	

10.8.3.3.2.4 Pluggable Module Association PDR

A plug association PDR exists per cable per cage of the device, as reflected from the topology file. For breakout cables, multiple cables may be associated with the same plug.

This PDR exists only if a module is plugged in and a cable is plugged in it. For a cable plug-in or plug-out event, a PDR change should be triggered through the NIC configuration change event.

Plug #n Entity Association PDR		
Container ID	1009 + n + PLDM_Offset*8	
Container Entity		
Entity Type	206 (SFP28 Module) 207 (SFP+ Module) 211 (QSFP28 Module) 212 (QSFP+ Module)	Plug
Entity Instance Number	1	
Container Entity Container ID	1039 + n + PLDM_Offset*8	
Association Type	Physical to Physical containment	
Contained Entity - Cable #1		
Entity Type	187	Cable
Entity Instance Number	1	
Container Entity Container ID	1009 + n + PLDM_Offset	
Contained Entity - Cable #2 - Breakout Only		
Entity Type	187	Cable
Entity Instance Number	2	
Container Entity Container ID	1009 + n + PLDM_Offset	
Contained Entity - Cable #3 - Breakout Only		
Entity Type	187	Cable
Entity Instance Number	3	
Container Entity Container ID	1009 + n + PLDM_Offset	
Contained Entity - Cable #4 - Breakout Only		
Entity Type	187	Cable
Entity Instance Number	4	
Container Entity Container ID	1009 + n + PLDM_Offset	

10.8.3.3.3 Logical Association PDRs

The PDRs are exposed in a high-level representation for the association PDRs.

10.8.3.3.3.1 Communication Channel Entity Association PDR

This PDR describes a link. It defines a channel that associates a logical port (MAC) with a pluggable module, and a specific cable in the module.

This PDR exists only if a module is plugged in and a cable is plugged into it, and the MAC is associated with a specific cable. For a cable plug-in or plug-out event, a PDR change should be triggered through the NIC configuration change event.

Channel #1 Entity Association		
Container ID	1059 + Channel# + PLDM_Offset*8	
Container Entity		
Entity Type	6	Communication Channel
Entity Instance Number	1 + PLDM_Offset	
Container Entity Container ID	100	NIC
Association Type	Logical to Physical containment	
Contained Entity - Ethernet Port		
Entity Type	300	Ethernet Port
Entity Instance Number	Port#	
Container Entity Container ID	1000 + PLDM_Offset	
Contained Entity - Cable		
Entity Type	187	Cable
Entity Instance Number	Cable# (is 1 for non-breakout modules)	Number of the cable within the plug
Container Entity Container ID	1009 + Module# + PLDM_Offset*8	

10.8.3.3.4 Composite State Sensor PDRs

10.8.3.3.4.1 NIC Composite State Sensor

NIC Composite State Sensor		
Entity Type	68	Add-in Card
Entity Instance Number	1	
Container Entity Container ID	0	
Terminus Handle	0	
Sensor ID	5	
Composite Sensor Count	4	
Sensor Type	1	Health State
Possible States	1=Normal, 3=Critical, 5=Upper_Non_Critical, 7=Upper Critical	
Sensor Type	15	Configuration
Possible States	1=Valid Configuration, 2=Invalid Configuration	
Sensor Type	16	Configuration Change
Possible States	1=Normal, 2=Change in Configuration	
Sensor Type	21	Thermal Trip
Possible States	1=Normal, 2=Thermal Trip	

10.8.3.3.4.2 Controller Composite State Sensor

Network Controller Composite State Sensor		
Entity Type	68	Add-in Card
Entity Instance Number	1 + PLDM_Offset	
Container Entity Container ID	100	
Terminus Handle	0	
Sensor ID	50	
Composite Sensor Count	4	
Sensor Type	1	Health State
Possible States	1=Normal, 3=Critical, 5=Upper_Non_Critical, 7=Upper Critical	
Sensor Type	15	Configuration
Possible States	1=Valid Configuration, 2=Invalid Configuration	
Sensor Type	16	Configuration Change
Possible States	1=Normal, 2=Change in Configuration	
Sensor Type	21	Thermal Trip
Possible States	1=Normal	

10.8.3.3.4.3 Plug Composite State Sensor

Plug #n Composite State Sensor		
Entity Type	189	Port
Entity Instance Number	1	
Container Entity Container ID	1039 + Module# + PLDM_Offset*8	
Terminus Handle	0	
Sensor ID	499 + n	
Composite Sensor Count	3	
Sensor Type	1	Health State
Possible States	0=Unknown, 1=Normal, 3=Critical, 4=Fatal, 5=Upper_Non_Critical	
Sensor Type	13	Presence
Possible States	1=Present, 2=Not_Present	
Sensor Type	21	Thermal Trip
Possible States	0=Unknown, 1=Normal, 2=Over-Temp Shutdown	

10.8.3.3.5 Simple State Sensors PDRs

10.8.3.3.5.1 Port Link State

Port #n Link State Sensor		
Entity Type	6	Communication Channel
Entity Instance Number	Port Number	The ordering number of the port reported in controller association PDR (Section 10.8.3.3.2.2)
Container Entity Container ID	1000	
Terminus Handle	0	
Sensor ID	699 + n	
Composite Sensor Count	1	
Sensor Type	33	Link State
Possible States	1=Connected, 2=Disconnected	

10.8.3.3.6 Numeric Sensors PDRs

10.8.3.3.6.1 Thermal Sensors

The parameters of the PDR for each sensor can be extracted through the GETNODEATTR DNL script on the Node Thermal Configuration Section.

Type	PDR Fields
uint32	recordHandle According to Section 10.8.3.1, "Sensors Numbering" (thermal sensors).
uint8	PDRHeaderVersion = 0x01
uint8	PDRType = 0x2 (Numeric Sensor PDR)
uint16	recordChangeNumber According to Section 10.8.2.3, "PDR Dynamic Changes Flow" .
uint16	dataLength = 0x43 (67)
uint16	PLDMTerminusHandle = 0x0001 + PLDM_Offset
uint16	sensorID According to Section 10.8.3.1, "Sensors Numbering" (thermal sensors).
uint16	entityType For Controller Sensor = 0x90 (Physical Network Controller (144)) For NIC Sensor = 0x44 (Physical Add-on card (68)) For Plug Sensor: 206 (SFP28 Module) 207 (SFP+ Module) 211 (QSFP28 Module) 212 (QSFP+ Module)
uint16	entityInstanceNumber For NIC Sensor: 1 For Controller Sensor: 1 + PLDM_Offset For Plug Sensor: 1
uint16	containerID According to the container of the thermal Sensor in Section 10.8.3.1, "Sensors Numbering" .
enum8	sensorInit = noInit (0x0) The Initialization Agent does not take any steps to initialize, enable or disable this particular sensor.
bool8	sensorAuxiliaryNamesPDR = FALSE Sensor does not have an associated Sensor Auxiliary Names PDR.
enum8	baseUnit = 0x2 (Degrees C).
sint8	unitModifier = 0x0 (No modifier)
enum8	rateUnit = None (0x0)
uint8	baseOEMUnitHandle = 0x0 (N/A)
enum8	auxUnit = None (0x0)
sint8	auxUnitModifier = 0x0
enum8	auxrateUnit = None (0x0)
enum8	Rel = 0x0 (N/A)
uint8	auxOEMUnitHandle = 0x0 (N/A)
bool8	isLinear = TRUE
enum8	sensorDataSize = 0x0 (uint8)

Type	PDR Fields
real32	resolution As read from resolution Low/Resolution High fields in Node Thermal Configuration.
real32	Offset As read from Offset Low/Offset High fields in Node Thermal Configuration Section. A constant value that is added in as part of the conversion process of converting a raw sensor reading to Units.
uint16	Accuracy As read from accuracy field in Node Thermal Configuration Section.
uint8	plusTolerance As read from Tolerance in Node Thermal Configuration Section.
uint8	minusTolerance As read from Tolerance in Node Thermal Configuration Section.
uint8	Hysteresis As read from Thermal Sensor Hysteresis in Node Thermal Configuration Section.
bitfield8	supportedThresholds Return 0x7 (<i>upperThresholdFatal</i> , <i>upperThresholdCritical</i> , <i>upperThresholdWarning</i>)
bitfield8	thresholdAndHysteresisVolatility Return 0x3 (PLDM subsystem power-up, Initialization Agent controller restart/update).
real32	stateTransitionInterval Return the internal firmware sensors polling interval (in real32). Current value is 1 second (0x3F800000).
real32	updateInterval Return the internal firmware sensors polling interval (in real32) - current value is 1 second (0x3F800000).
uint8	maxReadable As read from <i>maxReadable</i> in Node Thermal Configuration Section.
uint8	minReadable As read from <i>minReadable</i> in Node Thermal Configuration Section.
enum8	rangeFieldFormat = 0x2 (uint16)
bitfield8	rangeFieldSupport Return 0x2A (normalMax, criticalHigh, fatalHigh supported).
uint16	nominalValue = Return 0x0
uint16	normalMax As read from <i>normalMax</i> field in Node Thermal Configuration Section. Value is in degrees C, should be adapted according to the resolution field.
uint16	normalMin = Return 0x0
uint16	warningHigh As read from <i>warningHigh</i> field in Node Thermal Configuration Section or from module. Value is in °, should be adapted according to the resolution field.
uint16	warningLow = Return 0x0
uint16	criticalHigh As read from <i>criticalHigh</i> field in Node Thermal Configuration Section or from module. Value is in °C, should be adapted according to the resolution field.
uint16	criticalLow = Return 0x0
uint16	fatalHigh As read from <i>fatalHigh</i> field in Node Thermal Configuration Section. or from module. Value is in °C, should be adapted according to the resolution field Note: This threshold is not relevant for SFF Module's thermal sensor.
uint16	fatalLow = Return 0x0

Note: The Plug's values are read from the module itself, where the high thresholds used are mapped as follows:

SFF8636/SFF8472	PLDM (DSP0248)	Description
Warning	Warning	The reading is outside of normal expected operating range but the monitored entity is expected to continue to operate normally.
Alarm	Critical	The reading is outside of supported operating range. Monitored entities might operate abnormally, have transient failures, or propagate errors to other entities under this condition. Prolonged operation under this condition might result in degraded lifetime for the monitored entity.
N/A	Fatal	The reading is outside of rated operating range. Monitored entities might experience permanent failures or cause permanent failures to other entities under this condition.

10.8.3.3.6.2 Pluggable Module Power Sensors

Type	PDR Fields
uint32	recordHandle = 1699 + plug number (1..n) + PLDM_Offset*8
uint8	PDRHeaderVersion = 0x01
uint8	PDRType = 0x2 (Numeric Sensor PDR)
uint16	recordChangeNumber According to Section 10.8.2.3 , "PDR Dynamic Changes Flow".
uint16	dataLength = 0x47 (71)
uint16	PLDMTerminusHandle = 0x0001 + PLDM_Offset
uint16	sensorID = 399 + plug number (1..n)
uint16	entityType = 206 (SFP28 Module) / 207 (SFP+ Module) / 211 (QSFP28 Module) / 212 (QSFP+ Module)
uint16	entityInstanceNumber = 1
uint16	containerID = 1039 + Cage# + PLDM_Offset*8 (Connector)
enum8	sensorInit = noInit (0x0) The Initialization Agent does not take any steps to initialize, enable or disable this particular sensor.
bool8	sensorAuxiliaryNamesPDR = FALSE Sensor does not have an associated Sensor Auxiliary Names PDR
enum8	baseUnit = 0x7 (Watts) The reported value is the maximum power for the class.
sint8	unitModifier = -1 Tenth of watts.
enum8	rateUnit = 0x0 (None)
uint8	baseOEMUnitHandle = 0x0 (N/A)
enum8	auxUnit = None (0x0)
sint8	auxUnitModifier = 0x0
enum8	auxrateUnit = None (0x0)
enum8	Rel = 0x0 (N/A)
uint8	auxOEMUnitHandle = 0x0x (N/A)
bool8	isLinear = TRUE
enum8	sensorDataSize = 0x4 (uint32).

Type	PDR Fields
real32	Resolution = 0x0
real32	Offset = 0x0
uint16	Accuracy = 0x0
uint8	plusTolerance = 0x0
uint8	minusTolerance = 0x0
uint32	Hysteresis = 0x0 (sensor does not use hysteresis)
bitfield8	supportedThresholds = 0x0 (no threshold is supported)
bitfield8	thresholdAndHysteresisVolatility = 00000b (non-volatile)
real32	stateTransitionInterval = 0x3DCCCCD (100 ms - an upper limit on the firmware reaction time) How long the sensor device takes to do an <i>enabledState</i> change (worst case), in seconds.
real32	updateInterval = 0 (static value)
uint32	maxReadable = Actual power of plugged device == nominalValue == value reported by sensor.
uint32	minReadable = 0x0
enum8	rangeFieldFormat = 0x1 (uint8)
bitfield8	rangeFieldSupport = 0x1 (nominalValue field supported)
uint8	nominalValue Actual power of plugged device.
uint8	normalMax = 0x0
uint8	normalMin = 0x0
uint8	warningHigh = 0x0
uint8	warningLow = 0x0
uint8	criticalHigh = 0x0
uint8	criticalLow = 0x0
uint8	fatalHigh = 0x0
uint8	fatalLow = 0x0

10.8.3.3.6.3 Link Speed Sensors

Type	PDR Fields
uint32	recordHandle = 1299 + port number (1..n) + PLDM_Offset*8
uint8	PDRHeaderVersion = 0x01
uint8	PDRType = 0x2 (Numeric Sensor PDR)
uint16	recordChangeNumber According to Section 10.8.2.3 , "PDR Dynamic Changes Flow".
uint16	dataLength = 0x47 (71)
uint16	PLDMTerminusHandle = 0x0001 + PLDM Offset
uint16	sensorID = 99 + port number (1..n) + PLDM_Offset*8
uint16	entityType = 0x06 (Physical Port (6))

Type	PDR Fields
uint16	entityInstanceNumber Port number (the ordering number of the port reported in controller association PDR - Section 10.8.3.3.2.2 , "Controller Association PDR" this link refers to).
uint16	containerID = 1000 + PLDM Offset
enum8	sensorInit = noInit (0x0) The Initialization Agent does not take any steps to initialize, enable or disable this particular sensor.
bool8	sensorAuxiliaryNamesPDR = FALSE Sensor does not have an associated Sensor Auxiliary Names PDR.
enum8	baseUnit = 0x3C (Bits)
sint8	unitModifier = 0x6 // Mbps
enum8	rateUnit = 0x3 (per Second)
uint8	baseOEMUnitHandle = 0x0 (N/A)
enum8	auxUnit = None (0x0)
sint8	auxUnitModifier = 0x0
enum8	auxrateUnit = None (0x0)
enum8	Rel = 0x0 (N/A)
uint8	auxOEMUnitHandle = 0x0x (N/A)
bool8	isLinear = TRUE
enum8	sensorDataSize = 0x4 (uint32). Note: unit16 is currently enough to support 40G. Using uint32 to also cover 100G.
real32	Resolution = 0x0
real32	Offset = 0x0
uint16	Accuracy = 0x0
uint8	plusTolerance = 0x0
uint8	minusTolerance = 0x0
uint32	Hysteresis = 0x0 (sensor does not use hysteresis)
bitfield8	supportedThresholds = 0x0 (no threshold is supported)
bitfield8	thresholdAndHysteresisVolatility = 00000b (non-volatile)
real32	stateTransitionInterval = 0x3DCCCCCD (100 ms - an upper limit on the firmware reaction time) How long the sensor device takes to do an <i>enabledState</i> change (worst case), in seconds.
real32	updateInterval = 0x3D4CCCCD (50 ms - upper limit on the time from interrupt to event message)
uint32	maxReadable The maximum speed supported by the media in Mb/s 100G = 100,000 50G = 50,000 25G = 25,000 10G = 10,000 1G = 1,000 100 = 100
uint32	minReadable = 0x0 (returned when link is down)
enum8	rangeFieldFormat = 0x4 (uint32)
bitfield8	rangeFieldSupport = 0x1 (<i>nominalValue</i> field supported)

Type	PDR Fields
uint32	nominalValue The maximum speed supported by the media (in real32) 100G = 100,000 50G = 50,000 25G = 25,000 10G = 10,000 1G = 1,000
uint32	normalMax = 0x0
uint32	normalMin = 0x0

10.8.3.3.7 Redfish PDRs - Summary

Table 10-64 defines the PDRs that corresponding to the supported Schemas.

Table 10-64. PDRs corresponding to Supported Schema

Name	Type	Handle	ResourceID	Container	Profile	Section Reference
NetworkAdapter	Resource	4001	1 11	NetworkAdapters	ACD	10.8.3.3.8.1
NetworkInterface	Resource	4005	5	NetworkInterfaces	ACD	10.8.3.3.8.2
Ports	Resource	4010	10	1	ACD	10.8.3.3.8.3
NetworkDeviceFunctions	Resource	4020	20	1	ACD	10.8.3.3.8.4
Port	Resource	4100	100-107 110-117	10	ACD	10.8.3.3.8.5
NetworkDeviceFunction	Resource	4200	200-207 210-217	20	ACD	10.8.3.3.8.6
PCIeDevice	Resource	4003	3	System	ACD	10.8.3.3.8.7
PCIeFunctions	Resource	4030	30	3	ACD	10.8.3.3.8.8
PCIeFunction	Resource	4300	300-307	3	ACD	10.8.3.3.8.9
NetworkAdapterMetrics	Resource	4600	600	1	ACD	10.8.3.3.8.10
PortMetrics	Resource	4700	700-707	100-107	ACD	10.8.3.3.8.11
NetworkDeviceFunctionMetrics	Resource	4800	800-807	200-207	ACD	10.8.3.3.8.12
EthernetInterfaceCollection	Resource	4040	40	System	Other	10.8.3.3.9.1
EthernetInterface	Resource	4400	400-407 410-417	40	Other	10.8.3.3.9.2
NetworkAdapter.ResetSettingsToDefault	Action	3001	2001	-	Other	10.8.3.3.10.1
Port.Reset	Action	3002	-	-	ACD	10.8.3.3.10.2

10.8.3.3.8 ACD Profile PDRs and Links

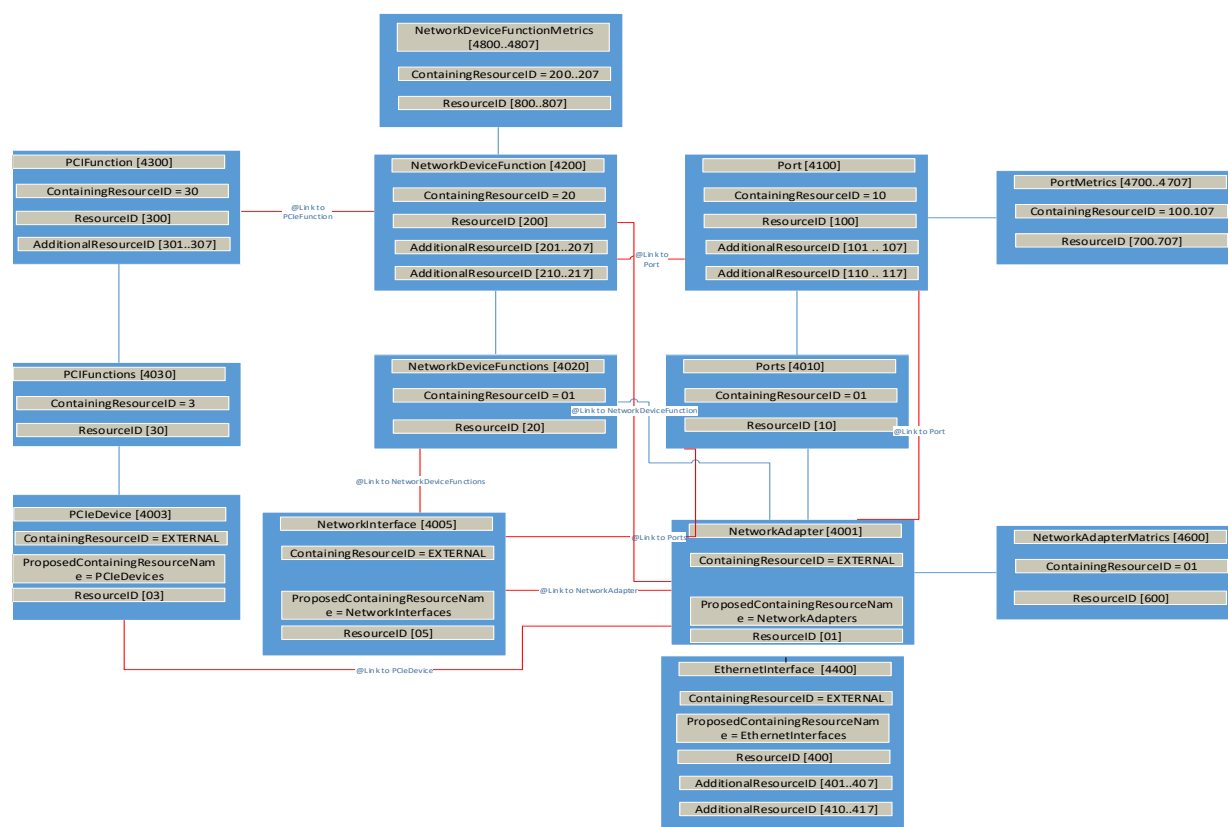


Figure 10-13. ACD Profile PDRs and Links

Note: The *recordHandle* field is part of the common PDR header and not part of the Redfish Resource PDR content. The *recordHandle* field appears in the following content tables as its value is different for each Redfish Resource PDR.

10.8.3.3.8.1 NetworkAdapter PDR

Type	Description	Value
uint32	recordHandle	4001
uint32	ResourceID	1
bitfield8	ResourceFlags	1 (is_device_root)
uint32	ContainingResourceID	0 (External)
uint16	ProposedContainingResourceLengthBytes	50
strUTF-8	ProposedContainingResourceName	"NetworkAdapterCollection.NetworkAdapterCollection\0"
uint16	SubURLengthBytes	
strUTF-8	SubURI	Null
uint16	AdditionalResourceIDCount	1
uint32	AdditionalResourceID [0]	11
uint16	AdditionalResourceSubURLengthBytes[0]	Length in bytes of SubURI.
strUTF-8	AdditionalResourceSubURI[0]	"Settings"
ver32	MajorSchemaVersion	0xF1F7F000
uint16	MajorSchemaDictionaryLengthBytes	Actual length of dictionary stored in NVM.
uint32	MajorSchemaDictionarySignature	32-bit CRC of the schema dictionary.
uint8	MajorSchemaNameLength	31
strUTF-8	MajorSchemaName	"NetworkAdapter. NetworkAdapter\0"
uint16	OEMCount	0

10.8.3.3.8.2 NetworkInterface PDR

Type	Description	Value
uint32	recordHandle	4005
uint32	ResourceID	5
bitfield8	ResourceFlags	1 (is_device_root)
uint32	ContainingResourceID	0 (External)
uint16	ProposedContainingResourceLengthBytes	54
strUTF-8	ProposedContainingResourceName	"NetworkInterfaceCollection.NetworkInterfaceCollection\0"
uint16	SubURLengthBytes	Length in bytes of SubURI
strUTF-8	SubURI	Null
uint16	AdditionalResourceIDCount	0
ver32	MajorSchemaVersion	0xF1F2F000
uint16	MajorSchemaDictionaryLengthBytes	Actual length of dictionary stored in NVM.
uint32	MajorSchemaDictionarySignature	32-bit CRC of the schema dictionary.
uint8	MajorSchemaNameLength	34
strUTF-8	MajorSchemaName	"NetworkInterface NetworkInterface\0"
uint16	OEMCount	0

10.8.3.3.3 Ports PDR

Type	Description	Value
uint32	recordHandle	4010
uint32	ResourceID	10
uint16	AdditionalResourceIDCount	0
bitfield8	ResourceFlags	4 (is_collection)
uint16	ProposedContainingResourceLengthBytes	1
strUTF-8	ProposedContainingResourceName	"\0"
uint16	SubURLengthBytes	Length in bytes of SubURI
strUTF-8	SubURI	"Ports"
uint32	ContainingResourceID	1
ver32	MajorSchemaVersion	0xFFFFFFFF
uint16	MajorSchemaDictionaryLengthBytes	Actual length of dictionary stored in NVM.
uint32	MajorSchemaDictionarySignature	32-bit CRC of the schema dictionary.
uint8	MajorSchemaNameLength	30
strUTF-8	MajorSchemaName	"PortCollection.PortCollection\0"
uint16	OEMCount	0

10.8.3.3.4 NetworkDeviceFunctions PDR

Type	Description	Value
uint32	recordHandle	4020
uint32	ResourceID	20
bitfield8	ResourceFlags	4 (is_collection)
uint32	ContainingResourceID	1
uint16	ProposedContainingResourceLengthBytes	1
strUTF-8	ProposedContainingResourceName	"\0"
uint16	SubURLengthBytes	Length in bytes of SubURI
strUTF-8	SubURI	"NetworkDeviceFunctions"
uint16	AdditionalResourceIDCount	0
ver32	MajorSchemaVersion	0xFFFFFFFF
uint16	MajorSchemaDictionaryLengthBytes	Actual length of dictionary stored in NVM.
uint32	MajorSchemaDictionarySignature	32-bit CRC of the schema dictionary.
uint8	MajorSchemaNameLength	64
strUTF-8	MajorSchemaName	"NetworkDeviceFunctionCollection.NetworkDeviceFunctionCollection\0"
uint16	OEMCount	0

10.8.3.3.8.5 Port PDR

Type	Description	Value
uint32	recordHandle	4100
uint32	ResourceID	100 + Ports[0]
bitfield8	ResourceFlags	2 (is_contained_in_collection)
strUTF-8	ProposedContainingResourceName	"\0"
uint16	SubURLengthBytes	Length in bytes of SubURI
strUTF-8	SubURI	"0"
uint16	AdditionalResourceIDCount	2*PortCount - 1
uint32	AdditionalResourceID [0]	100 + Ports[1]
uint16	AdditionalResourceSubURLengthBytes[0]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[0]	"1"
uint32	...	...
uint32	AdditionalResourceID [PortCount - 2]	100 + Ports[PortCount - 1]
uint16	AdditionalResourceSubURLengthBytes[PortCount - 2]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[PortCount - 2]	"n"
uint32	AdditionalResourceID [PortCount - 1]	110 + Ports[0]
uint16	AdditionalResourceSubURLengthBytes[PortCount - 1]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[PortCount - 1]	"0/Settings"
uint32	...	...
uint32	AdditionalResourceID [2*PortCount - 2]	110 + Ports[PFCCount-1]
uint16	AdditionalResourceSubURLengthBytes[2*PortCount - 2]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[2*PortCount - 2]	"n/Settings"
ver32	MajorSchemaVersion	0xF1F6F000
uint16	MajorSchemaDictionaryLengthBytes	Actual length of dictionary stored in NVM.
uint32	MajorSchemaDictionarySignature	32-bit CRC of the schema dictionary.
uint8	MajorSchemaNameLength	10
strUTF-8	MajorSchemaName	"Port.Port\0"
uint16	OEMCount	0

10.8.3.3.8.6 NetworkDeviceFunction PDR

Type	Description	Value
uint32	recordHandle	4200
uint32	ResourceID	200 + PFs[0]
bitfield8	ResourceFlags	2 (is_contained_in_collection)
uint32	ContainingResourceID	20
uint16	ProposedContainingResourceLengthBytes	1
strUTF-8	ProposedContainingResourceName	"\0"
uint16	SubURLengthBytes	Length in bytes of SubURI
strUTF-8	SubURI	"0"
uint16	AdditionalResourceIDCount	2*ExposedPFs-1
uint32	AdditionalResourceID [0]	200 + PFs[1]
uint16	AdditionalResourceSubURLengthBytes[0]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[0]	"1"
uint32	...	...
uint32	AdditionalResourceID [ExposedPFs - 2]	200 + PFs[ExposedPFs - 1]
uint16	AdditionalResourceSubURLengthBytes[ExposedPFs - 2]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[ExposedPFs - 2]	"n"
uint32	AdditionalResourceID [ExposedPFs - 1]	210 + PFs[0]
uint16	AdditionalResourceSubURLengthBytes[ExposedPFs - 1]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[ExposedPFs - 1]	"0/Settings"
uint32	...	...
uint32	AdditionalResourceID [2*ExposedPFs - 2]	210 + PFs[ExposedPFs-1]
uint16	AdditionalResourceSubURLengthBytes[2*ExposedPFs - 2]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[2*ExposedPFs - 2]	"n"/Settings"
ver32	MajorSchemaVersion	0xF1F6F000
uint16	MajorSchemaDictionaryLengthBytes	Actual length of dictionary stored in NVM.
uint32	MajorSchemaDictionarySignature	32-bit CRC of the schema dictionary.
uint8	MajorSchemaNameLength	44
strUTF-8	MajorSchemaName	"NetworkDeviceFunction.NetworkDeviceFunction\0"
uint16	OEMCount	0

10.8.3.3.8.7 PCIeDevice PDR

Type	Description	Value
uint32	recordHandle	4003
uint32	ResourceID	3
bitfield8	ResourceFlags	1 (is_device_root)
uint32	ContainingResourceID	0 (External)
uint16	ProposedContainingResourceLengthBytes	42
strUTF-8	ProposedContainingResourceName	"PCIeDeviceCollection.PCIeDeviceCollection\ 0"
uint16	SubURLengthBytes	Length in bytes of SubURI
strUTF-8	SubURI	null
uint16	AdditionalResourceIDCount	0
uint32	MajorSchemaVersion	0xF1F4F000
uint16	MajorSchemaDictionaryLengthBytes	Actual length of dictionary stored in NVM
uint32	MajorSchemaDictionarySignature	32-bit CRC of the schema dictionary
uint8	MajorSchemaNameLength	22
strUTF-8	MajorSchemaName	"PCIeDevice.PCIeDevice\0"
uint16	OEMCount	0

10.8.3.3.8.8 PCIeFunctions PDR

Type	Description	Value
uint32	recordHandle	4030
uint32	ResourceID	30
bitfield8	ResourceFlags	4 (is_collection)
uint32	ContainingResourceID	1
uint16	ProposedContainingResourceLengthBytes	1
strUTF-8	ProposedContainingResourceName	"\0"
uint16	SubURLengthBytes	Length in bytes of SubURI
strUTF-8	SubURI	"PCIeFunctions"
uint16	AdditionalResourceIDCount	0
ver32	MajorSchemaVersion	0xFFFFFFFF
uint16	MajorSchemaDictionaryLengthBytes	Actual length of dictionary stored in NVM.
uint32	MajorSchemaDictionarySignature	32-bit CRC of the schema dictionary.
uint8	MajorSchemaNameLength	46
strUTF-8	MajorSchemaName	"PCIeFunctionCollection.PCIeFunctionCollection\0"
uint16	OEMCount	0

10.8.3.3.8.9 PCIeFunction PDR

Type	Description	Value
uint32	recordHandle	4300
uint32	ResourceID	300 + PFs[0]
bitfield8	ResourceFlags	2 (is_contained_in_collection)
uint32	ContainingResourceID	30
uint16	ProposedContainingResourceLengthBytes	1
strUTF-8	ProposedContainingResourceName	"\0"
uint16	SubURLengthBytes	Length in bytes of SubURI
strUTF-8	SubURI	"/0"
uint16	AdditionalResourceIDCount	PFCCount - 1
uint32	AdditionalResourceID [0]	300 + PFs[1]
uint16	AdditionalResourceSubURLengthBytes[0]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[0]	"/1"
uint32	...	...
uint32	AdditionalResourceID [PortCount - 2]	300 + PFs[PFCCount - 1]
uint16	AdditionalResourceSubURLengthBytes[PortCount - 2]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[PortCount - 2]	"/n"
ver32	MajorSchemaVersion	0xF1F2F100
uint16	MajorSchemaDictionaryLengthBytes	Actual length of dictionary stored in NVM.
uint32	MajorSchemaDictionarySignature	32-bit CRC of the schema dictionary.
uint8	MajorSchemaNameLength	26
strUTF-8	MajorSchemaName	"PCIeFunction.PCIeFunction\0"
uint16	OEMCount	0

10.8.3.3.8.10 NetworkAdapterMetrics PDR

Type	Description	Value
uint32	recordHandle	4600
uint32	ResourceID	600
bitfield8	ResourceFlags	2 (is_contained_in_collection)
uint32	ContainingResourceID	1
uint16	ProposedContainingResourceLengthBytes	1
strUTF-8	ProposedContainingResourceName	"\0"
uint16	SubURLengthBytes	Length in bytes of SubURI
strUTF-8	SubURI	"Metrics"
uint16	AdditionalResourceIDCount	0
ver32	MajorSchemaVersion	0xF1F0F000
uint16	MajorSchemaDictionaryLengthBytes	Actual length of dictionary stored in NVM

Type	Description	Value
uint32	MajorSchemaDictionarySignature	32-bit CRC of the schema dictionary
uint8	MajorSchemaNameLength	44
strUTF-8	MajorSchemaName	"NetworkAdapterMetrics.NetworkAdapterMetrics\0"
uint16	OEMCount	0

10.8.3.3.8.11 PortMetrics PDR

Type	Description	Value
uint32	recordHandle	4700 + Port#
uint32	ResourceID	700 + Port#
bitfield8	ResourceFlags	2 (is_contained_in_collection)
uint32	ContainingResourceID	100 + Port#
uint16	ProposedContainingResourceLengthBytes	1
strUTF-8	ProposedContainingResourceName	"\0"
uint16	SubURLengthBytes	Length in bytes of SubURI
strUTF-8	SubURI	"Metrics"
uint16	AdditionalResourceIDCount	0
ver32	MajorSchemaVersion	0xF1F2F000
uint16	MajorSchemaDictionaryLengthBytes	Actual length of dictionary stored in NVM
uint32	MajorSchemaDictionarySignature	32-bit CRC of the schema dictionary
uint8	MajorSchemaNameLength	24
strUTF-8	MajorSchemaName	"PortMetrics.PortMetrics\0"
uint16	OEMCount	0

Note: There is a PortMetrics PDR per Port.

10.8.3.3.8.12 NetworkDeviceFunctionMetrics PDR

Type	Description	Value
uint32	recordHandle	4800 + PF#
uint32	ResourceID	800 + PF#
bitfield8	ResourceFlags	2 (is_contained_in_collection)
uint32	ContainingResourceID	200 + PF#
uint16	ProposedContainingResourceLengthBytes	1
strUTF-8	ProposedContainingResourceName	"\0"
uint16	SubURLengthBytes	Length in bytes of SubURI
strUTF-8	SubURI	"Metrics"
uint16	AdditionalResourceIDCount	0
ver32	MajorSchemaVersion	0xF1F1F000

Type	Description	Value
uint16	MajorSchemaDictionaryLengthBytes	Actual length of dictionary stored in NVM
uint32	MajorSchemaDictionarySignature	32-bit CRC of the schema dictionary
uint8	MajorSchemaNameLength	58
strUTF-8	MajorSchemaName	"NetworkDeviceFunctionMetrics.NetworkDeviceFunctionMetrics\0"
uint16	OEMCount	0

Note: There is a NetworkDeviceFunctionMetrics PDR per Port.

10.8.3.3.9 Non-ACD PDRs

There are several schemas that are beyond the ACD profile definition. Figure 10-14 defines those PDRs. The supported fields are based on the OCP Baseline Hardware Management profile v1_0_0.



Figure 10-14. Non-ACD PDRs

10.8.3.3.9.1 EthernetInterfaceCollection PDR

Type	Description	Value
uint32	recordHandle	4040
uint32	ResourceID	40
bitfield8	ResourceFlags	5 (is_device_root is_collection)
uint32	ContainingResourceID	0 (External)
uint16	ProposedContainingResourceLengthBytes	6
strUTF-8	ProposedContainingResourceName	"System\0"
uint16	SubURLengthBytes	Length in bytes of SubURI
strUTF-8	SubURI	Null
uint16	AdditionalResourceIDCount	0
ver32	MajorSchemaVersion	0xFFFFFFFF
uint16	MajorSchemaDictionaryLengthBytes	Actual length of dictionary stored in NVM.
uint32	MajorSchemaDictionarySignature	32-bit CRC of the schema dictionary.
uint8	MajorSchemaNameLength	28
strUTF-8	MajorSchemaName	"EthernetInterfaceCollection. EthernetInterfaceCollection"
uint16	OEMCount	0

10.8.3.3.9.2 EthernetInterface PDR

Type	Description	Value
uint32	recordHandle	4400
uint32	ResourceID	400 + Ports[0]
bitfield8	ResourceFlags	2 (is_contained_in_collection)
uint32	ContainingResourceID	40
uint16	ProposedContainingResourceLengthBytes	1
strUTF-8	ProposedContainingResourceName	"\0"
uint16	SubURLengthBytes	Length in bytes of SubURI
strUTF-8	SubURI	"0"
uint16	AdditionalResourceIDCount	2*PortCount-1
uint32	AdditionalResourceID [0]	400 + Ports[1]
uint16	AdditionalResourceSubURLengthBytes[0]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[0]	"1"
uint32	...	...
uint32	AdditionalResourceID [PortCount - 2]	400 + Ports[PortCount - 1]
uint16	AdditionalResourceSubURLengthBytes[PortCount - 2]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[PortCount - 2]	"n"
uint32	AdditionalResourceID [PortCount - 1]	410 + Ports[0]
uint16	AdditionalResourceSubURLengthBytes[PortCount - 1]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[PortCount - 1]	"0/Settings"
uint32	...	...
uint32	AdditionalResourceID [2*PortCount - 2]	410 + Ports[PortCount-1]
uint16	AdditionalResourceSubURLengthBytes[PortCount - 2]	Length in bytes of SubURI
strUTF-8	AdditionalResourceSubURI[PortCount - 2]	"n"/Settings"
ver32	MajorSchemaVersion	0xF1F4F100
uint16	MajorSchemaDictionaryLengthBytes	Actual length of dictionary stored in NVM.
uint32	MajorSchemaDictionarySignature	32-bit CRC of the schema dictionary.
uint8	MajorSchemaNameLength	18
strUTF-8	MajorSchemaName	"EthernetInterface.EthernetInterface"
uint16	OEMCount	0

10.8.3.3.10 RDE Action PDRs

The actions currently supported are the ResetSettingsToDefault action PDR related to NetworkAdapter resource, and the Reset action PDR related to the Port resource.

See DSP0248 for the PDR definition.

10.8.3.3.10.1 NetworkAdapter.ResetSettingsToDefault Action PDR

Type	Description	Value
uint32	recordHandle	3001
...	[the rest of Common PDR Header]	...
uint8	ActionPDRIndex	0
uint16	RelatedResourceCount	1
uint32	RelatedResourceID [0]	1
uint8	ActionCount	1
uint8	ActionNameLengthBytes [0]	23
utf8string	ActionName [0]	"ResetSettingsToDefault\0"
uint8	ActionPathLengthBytes [0]	ActionPath length in bytes (46).
utf8string	ActionPath [0]	"Actions/ NetworkAdapter.ResetSettingsToDefault\0"

10.8.3.3.10.2 Port.Reset Action PDR

Type	Description	Value
uint32	recordHandle	3002
?...	[The rest of Common PDR Header]	...
uint8	ActionPDRIndex	0
uint16	RelatedResourceCount	Number of ports N
uint32	RelatedResourceID [0]	100 (Port[0]'s resource ID)
...	...	...
uint32	RelatedResourceID [N-1]	100 + N - 1 (Last Port's resource ID)
uint8	ActionCount	1
uint8	ActionNameLengthBytes [0]	6
utf8string	ActionName [0]	"Reset\0"
uint8	ActionPathLengthBytes [0]	ActionPath length in bytes (19).
utf8string	ActionPath [0]	"Actions/Port.Reset\0"

10.8.4 PLDM Firmware Update Commands

The firmware should support the following PLDM commands and responses according to DSP0267.

Table 10-65. PLDM for Firmware Update Command Codes

Command	Command Code	Command Requestor (Initiator)	Supported	Section Reference
QueryDeviceIdentifiers	0x01	BMC	Yes	10.8.4.2.1
GetFirmwareParameters	0x02	BMC	Yes	10.8.4.2.2
RequestUpdate	0x10	BMC	Yes	10.8.4.2.3
GetPackageData	0x11	Firmware	Yes	10.8.4.1.1
GetDeviceMetaData	0x12	BMC	No	---
PassComponentTable	0x13	BMC	Yes	10.8.4.2.4
UpdateComponent	0x14	BMC	Yes	10.8.4.2.5
RequestFirmwareData	0x15	Firmware	Yes	10.8.4.1.2
TransferComplete	0x16	Firmware	Yes	10.8.4.1.3
VerifyComplete	0x17	Firmware	Yes	10.8.4.1.4
ApplyComplete	0x18	Firmware	Yes	10.8.4.1.5
GetMetaData	0x19	Firmware	No	---
ActivateFirmware	0x1A	BMC	Yes	10.8.4.2.6
GetStatus	0x1B	BMC	Yes	10.8.4.2.7
CancelUpdateComponent	0x1C	BMC	Yes	10.8.4.2.8
CancelUpdate	0x1D	BMC	Yes	10.8.4.2.9

10.8.4.1 Firmware-to-BMC Commands

This section describes the commands that are sent by firmware to BMC. Refer to DSP0267 for full descriptions of each command and its fields.

Upon error code or timeout, in response firmware should act as follows:

- Implement “6.3.2 Requirements for Requesters” as specified in DSP0240.
 - Number of request retries PN1.
 - Request-to-response time PT1.
 - Time out waiting for a response PT2.
- In addition, firmware should implement the state transition for error or timeout, as explained in “8.2 State machine” in DSP0267.
- For an error that does not allow firmware to continue the update and firmware actions are not described in “8.2 State machine” in DSP0267, firmware does one of the following:
 - Cancel Update (return to IDLE) for an error in the following states:
 - IDLE
 - LEARN
 - ACTIVATE

— Cancel Component Update (return to READY XFER):

- READY XFER
- DOWNLOAD
- VERIFY
- APPLY

10.8.4.1.1 GetPackageData (0x11)

The package data includes the following fields:

- Byte [0].Bit 0 = PFA Preserve (default mode). Should be set to zero.
- Byte [0].Bit 1 = Reserved (was O-ROM version to be set in VPD).
- Byte [0].Bits [7:2] = Reserved
- TLV as follows:
 - Type – 2 bytes:
 - CURRENT_GFID – 0x1
 - Length – 2 bytes = 36 words
 - Value:
 - 17:0 = Current GFID
 - 35:18 = Original GFID

GFID length and value: Same as in [Section 6.3.5.2](#) and [Section 6.3.48](#).

It is assumed that GFID is unique per OEM thus should not change during the update.

Current GFID is the GFID of the currently running image planned to update. Original GFID is the basic image from which this map is derived.

- A set of TLVs as needed as follows:
 - Type – 2 bytes:
 - TLV + Offset update - 0x10 // used to update TLVs
 - Length – 2 bytes:
 - According to size of value (in words)
 - Value
 - uint16: TLV number
 - uint16: offset in words (offset 0 is the beginning of the TLV data)
 - uint16: Data length in words
 - Variable: Data
- A set of TLVs as needed as follows:
 - Type – 2 bytes:
 - VPD Key update - 0x11 // Used to update VPD entries
 - Length - 2 bytes (According to size of value (in words))

— Value:

- uint16: VPD key (ascii - e.g. V2 = 0x5632 (ASCII('V'), ASCII('2')))
- uint16: Data length (in bytes) // should match current VPD entry size - error otherwise
Note: If Data Length is odd, add a pad byte at end of data to make the TLV word aligned.
- Variable: Data

Note: The TLVs can be in any order.

10.8.4.1.2 RequestFirmwareData (0x15)

For the FD to retrieve a section of a component image, the FD sends *RequestFirmwareData* request message to the UA, specifying its offset and length. The UA sends a response message that includes the component image portion specified by the offset and length from the request message. See DSP0267, Chapter 11.6.

Implementation note: Firmware should minimize the number of NVM Update commands by doing 4 KB updates.

10.8.4.1.3 TransferComplete (0x16)

The FD sends *TransferComplete* command to the UA once the FD has transferred all the data for the component image or determines the transfer has failed. See DSP0267, Chapter 11.7.

10.8.4.1.4 VerifyComplete (0x17)

Firmware performs a validation check against the component image that was received. See DSP0267, Chapter 11.8.

Firmware performs the same verification check as performed today during the NVM/S-RAM/O-ROM update using the NVM Update admin commands. See [Section 3.3.5](#).

Firmware should also verify that the parameters received before the download match the downloaded components.

10.8.4.1.5 ApplyComplete (0x18)

As firmware has already completed the component image transfer into the storage location, it should only return the Activation method:

- [5] AC power cycle = POR
- [4] DC power cycle = PCIe Reset
- [3] System reboot = In-band Reset
- [2] Medium-specific reset = EMPR
- [1] Self-Contained reset = EMPR is performed by firmware upon *ActivateFirmware*
- [0] Automatic = EMPR is performed by firmware as the Apply completes, or as download completes if the FD performs an auto-apply

Command Parameter	Description
ComponentActivationMethodsModification	"DC power cycle" or "Self-Contained reset" based on the decision described in Section 10.8.5.6 .

10.8.4.2 BMC-to-Firmware Commands

This section describes the firmware responses for commands that arrived from the BMC.

Refer to DSP0267 for full description of each command and its fields.

10.8.4.2.1 QueryDeviceIdentifiers (0x01)

This command is used by the UA to obtain the firmware identifiers for the FD. The FD must provide a response message to this command in all states, including IDLE. See DSP0267, Chapter 10.1.

Response Parameter	Description
DescriptorCount	Should be set to 4.
Descriptors	Refer to Section 10.8.5.2.1 for FD <i>InitialDescriptorType</i> , <i>InitialDescriptorLength</i> , <i>InitialDescriptorData</i> .

10.8.4.2.2 GetFirmwareParameters (0x02)

The UA sends *GetFirmwareParameters* command to acquire the component details, such as classification types and corresponding versions of the FD. The FD must provide a response message to this command in all states, including IDLE. See DSP0267, Chapter 10.2.

Response Parameter	Description
CapabilitiesDuringUpdate	Set the following bits: Bit 4 = 0 – No host OS environment restriction for update mode. Bit 3 = 1 – Firmware Device can support a partial update, whereby a package that contains a component image set that is a subset of all components currently residing on the FD, can be transferred. Bit 2 = 1 – Device host functionality will be reduced, perhaps becoming inaccessible, during Firmware Update. Bit 1 = 0 – Device can have component updated again without exiting update mode and restarting transfer via <i>RequestUpdate</i> command. Bit 0 = 0 – Device will revert to previous component image upon a failure, timeout, or cancellation of the transfer.
ComponentCount	Set to 3 (NVM, OROM, Netlist).
ActiveComponentImageSetVersionStringType	Refer to <i>ComponentImageSetVersionStringType</i> in Section 10.8.5.2.1 .
PendingComponentImageSetVersionStringType	Same as for active set, but for pending component.
ActiveComponentImageSetVersionString	Refer to <i>ComponentImageSetVersionString</i> in Section 10.8.5.2.1 .
PendingComponentImageSetVersionString	Same format as for active set.
ComponentClassification	Refer to <i>ComponentClassification</i> in Section 10.8.5.2.1 .
ComponentIdentifier	Refer to <i>ComponentIdentifier</i> in Section 10.8.5.2.1 .
ComponentClassificationIndex	Not used. Set to 0.
ActiveComponentComparisonStamp	Refer to <i>ComponentComparisonStamp</i> in Section 10.8.5.2.1 .
ActiveComponentVersionStringType	Refer to <i>ComponentVersionStringType</i> in Section 10.8.5.2.1 .
ActiveComponentReleaseDate	Refer to <i>ComponentReleaseDate</i> in Section 10.8.5.2.1 .

Response Parameter	Description
PendingComponentComparisonStamp	Same format as for active component.
PendingComponentVersionStringType	Same as for active component.
PendingComponentReleaseDate	Same format as for active component.
ComponentActivationMethods	"DC power cycle" or "Self-Contained reset".
CapabilitiesDuringUpdate	Bit 0 = 0 – Firmware Device executes an operation during the APPLY state, that includes migrating the new component image to its final non-volatile storage destination.
ActiveComponentVersionString	Refer to <i>ComponentVersionString</i> in Section 10.8.5.2.1 .
PendingComponentVersionString	Same as for active component.

10.8.4.2.3 RequestUpdate (0x10)

This is the first PLDM command to initiate a firmware update for an FD. The FD must enter update mode if that command response indicates success. See DSP0267, Chapter 11.1.

Firmware should verify the parameters that have been received and return errors in cases when it does not match.

Firmware takes OWNERSHIP over NVM. In any case of return to IDLE state due to:

- Timeout
- Reset
- Error

After already being in update mode, firmware should release the OWNERSHIP of the NVM.

Command Parameter	Description
NumberOfComponents	Up to "3". Should be $\leq$ <i>ComponentCount</i> in <i>GetFirmwareParameters</i> .
PackageDataLength	Should be > 0 . See Section 10.8.4.1.1 .

Response Parameter	Description
FirmwareDeviceMetaDataLength	Set to 0, as <i>GetMetaData</i> is not supported.

10.8.4.2.4 PassComponentTable (0x13)

The *PassComponentTable* command is used to pass component information to the FD after the FD enters update mode. The *PassComponentTable* command contains the component information table for a specific component including *ComponentClassificationIndex*, *ComponentClassification*, and version details.

If the firmware update package contains more than one component, multiple *PassComponentTable* commands must be sent by the UA (one for each component). The UA must pass the component table for all applicable components listed in the firmware package header in ascending order of index. See DSP0267, Chapter 11.4.

Firmware verifies the parameters that have been received and returns an error in case they do not match.

Command Parameter	Description
ComponentClassification	Refer to <i>ComponentClassification</i> at in Section 10.8.5.2.1 .
ComponentIdentifier	Refer to <i>ComponentIdentifier</i> in Section 10.8.5.2.1 .
ComponentComparisonStamp	Refer to <i>ComponentComparisonStamp</i> in Section 10.8.5.2.1 .
ComponentVersionStringType	Refer to <i>ComponentVersionStringType</i> in Section 10.8.5.2.1 .
ComponentVersionString	Refer to <i>ComponentVersionString</i> in Section 10.8.5.2.1 .

10.8.4.2.5 UpdateComponent (0x14)

The UA sends the *UpdateComponent* command to request updating a specific firmware component. See DSP0267, Chapter 11.5.

Command Parameter	Description
UpdateOptionFlags	Bit 0 = Request Force Update of component – Can be used to inform the FD device to perform a transfer even if the component has a lower or equal component comparison stamp, or version string, than what is installed. The UA sets this bit for any component that has the force update bit set in the <i>ComponentOptions</i> field of the package header. Additionally, the UA could set the bit as instructed by commands used to provide the update package to the UA (these commands are out of scope for this specification).

If Bit 0 is set, firmware should allow downgrade only up to security version.

10.8.4.2.6 ActivateFirmware (0x1A)

After all firmware components in the FD have been transferred and applied, the UA sends this command to inform the FD to prepare all successfully-applied components to become active at the next activation.

The UA can also request activation of all components that have an activation method of “Self-Contained”. See DSP0267, Chapter 11.11.

Command Parameter	Description
SelfContainedActivationRequest	Self contained activate is not supported.

Response Parameter	Description
EstimatedTimeForSelfContainedActivation	Should be set to 0 if no self-contained components.

10.8.4.2.7 GetStatus (0x1B)

The UA sends this command to acquire the status of the FD. See DSP0267, Chapter 11.12.

Response Parameter	Description
ProgressPercent	IDLE – 0% LEARN COMPONENTS – 0% READY XFER – 0% DOWNLOAD – $100\% * \text{num\_downloaded} / \text{total\_num\_of\_comp}$ VERIFY – $100\% * \text{num\_verified} / \text{total\_num\_of\_comp}$ APPLY – $100\% * \text{num\_applied} / \text{total\_num\_of\_comp}$ ACTIVATE – 0%

10.8.4.2.8 CancelUpdateComponent (0x1C)

During the firmware component transfer process, the UA can send this command to the FD. The FD, upon receiving this command, must stop sending *RequestFirmwareData* commands to the UA, and cancel the current component update procedure. The FD controller must transition to the READY XFER state of update mode and be ready to accept another *UpdateComponent* command. The UA can attempt to resend the same component image to the UA. See DSP0267, Chapter 11.13.

See DSP0267, Chapter 8.2, "State Machine".

Even though it is not required by DSP0267, it is expected that firmware get back to normal activity using the currently active component.

10.8.4.2.9 CancelUpdate (0x1D)

This command signals to the FD that it should exit from update mode even if activation is required to begin operating at the new firmware level. The UA should always attempt to complete the transfer of all components and use this command only if it determines that there is no other method to continue with the transfer process. See DSP0267, Chapter 11.14.

See DSP0267, Chapter 8.2, "State Machine".

Even though it is not required by DSP0267, it is expected that firmware get back to normal activity using the currently active components.

Response Parameter	Description
NonFunctioningComponentIndication	Always False meaning that all components will be functional.
NonFunctioningComponentBitmap	All zeros.

10.8.5 PLDM Firmware Update Flow

This section describes the implementation of DSP0267 - PLDM firmware update protocol.

10.8.5.1 Update Components

The firmware update over PLDM can be done to the following NVM components:

- NVM
- OROM
- Netlist

Note: The Shadow RAM is a component that is not updated through PLDM, but through specific commands upon demand. Shadow RAM might be considered as a component in the future.

10.8.5.2 Firmware Update Package

This section describes the fields of the Firmware Update Package that are relevant for this implementation.

The comparison algorithm for topology netlist is:

CN = Current Netlist

NN = New Netlist

CV = CustomerNetlistVersion

BRV = BaseReleaseVersion

If (CN.BaseReleaseVersion.Type <> NN.BaseReleaseVersion.Type), then Error (not matching Netlist).

If (CN.OEM IANA <> NN.OEM IANA) then Error (not matching OEM).

Else according to the following table:

BRV."major.minor"	CV	Result	Notes
CN = NN	CN = NN	Same version	Everything is equal so it is the same version.
CN < NN	Don't care	Upgrade	If the BRV is higher, allow upgrade regardless of CV setting.
CN = NN	CN < NN	Upgrade	If the BRV is equal, allow upgrade when CV setting is higher in the New Netlist.
CN > NN	Don't care	Downgrade	If the BRV is lower, allow downgrade regardless of CV setting.
CN = NN	CN > NN	Downgrade	If the BRV is equal, allow downgrade when CV setting is lower in the New Netlist.

Table 10-67. Firmware Device

Type	Name	Format/Value	Mandatory
uint16	InitialDescriptorType Indicates the type of the Initial descriptor. The initial descriptor for a device must be defined by one of the following (PCI Vendor ID, IANA Enterprise ID, UUID, PnP Vendor ID, or ACPI Vendor ID). If the FD uses Vendor Defined values as part of its implementation of this specification (for example to provide a vendor defined error code or component classification), the initial descriptor must be set to either PCI Vendor ID or IANA Enterprise ID.	PCI Vendor ID = 0x0000 Length = 2 Bytes	Yes
Variable	InitialDescriptorData Payload containing the identifier value for the initial descriptor.	0x8086	Yes
uint16	AdditionalDescriptorType0	PCI Device ID	Yes
Variable	AdditionalDescriptorData0	See Section 9.2.2.2 . Note: All the descriptor fields are taken from the config space of PF0.	Yes
uint16	AdditionalDescriptorType1	PCI Subsystem Vendor ID	Yes
Variable	AdditionalDescriptorData1	See Section 9.2.2.10 .	Yes
uint16	AdditionalDescriptorType2	PCI Subsystem ID	Yes
Variable	AdditionalDescriptorData2	See Section 9.2.2.11 .	Yes
uint16	AdditionalDescriptorType3	PCI Revision ID	No
Variable	AdditionalDescriptorData3	See Section 9.2.2.5 .	No

Table 10-68. Component Set

Type	Name	Format/Value	Mandatory
enum8	ComponentImageSetVersionStringType ASCII, UTF-8/16/16LE/16BE, Unknown	ASCII – 1	Yes
Variable	ComponentImageSetVersionString Up to 255 bytes	See Table 10-66 OEM Gen ComponentSetVersionString. When written, the <i>ComponentImageSetVersionString</i> is stored in PFV TLV #0x127.	Yes

Table 10-69. Component

Type	Name	Format/Value	Mandatory
enum8	ComponentVersionStringType ASCII, UTF-8/16/16LE/16BE, Unknown	ASCII – 1 NVM Length = 18 O-ROM Length = 18 Netlist length = 50	Yes
Variable	ComponentVersionString Up to 255 bytes	See Table 10-66 .	Yes
uint16	ComponentClassification 0x000A - Firmware, 0x000B - BIOS/FCODE, 0x0006 - Firmware/BIOS, 0x8000-0xFFFF – Vendor defined	0x000A - Firmware	Yes
uint16	ComponentIdentifier FD vendor selected unique value to distinguish between component images.	NVM lad_module_id = 0x6 O-ROM lad_module_id = 0x5 Topology Netlist Module ID = 0x8	Yes
uint8	ComponentClassificationIndex Used to distinguish identical components that have the same classification and identifier that can use the same component image but the images are stored in different locations in the FD.	Not used.	No
uint32	ComponentComparisonStamp When <i>ComponentOptions</i> bit 1 is set, this field must contain a FD vendor selected value to use as a comparison value in determining if a firmware component is down-level or up-level. For the same component identifier, the greater of two component comparison stamps is considered up-level compared to the other when performing an unsigned integer comparison. FD vendors should choose the value for the comparison stamp in a manner that permits interim component versions, such as patch releases. For example, a value for this field can follow the format of <i>MajorMinorRevisionPatch</i> where each subfield has a range of 0x00 to 0xFF.	See Table 10-66 .	No
ASCII[8]	ComponentReleaseDate Optional. Firmware returns this information upon <i>GetFirmwareParameters</i> command. Note: It is not part of Component Image information that is passed by BMC to FD. If supported, it is assumed to be part of the component image.	Not used.	No

10.8.5.3 Firmware Flow for Section Update

Table 10-70 includes the commands and the firmware action for each command for any bank update.

Table 10-70. Action Upon Reception of Commands - NVM Section

BMC Command	Firmware Device Command	Firmware Device Action	Remarks	Section Reference
RequestUpdate		Take ownership on NVM. Respond success or other error code. Note: The firmware holds ownership of the NVM for 15 minutes. After that, the update process is aborted and NVM ownership is released.	First PLDM command to initiate a firmware update for an FD. Same as Section 10.11.2.7.3 From the standard: "The FD must then enter an update mode that no longer permits another update request until the UA finishes or cancels the firmware update."	10.8.4.2.3
	GetPackageData			10.8.4.1.1
PassComponentTable		Respond success or other error code. Perform the following checking: Check GFID as described in Section 3.3.5.6.1 . Check security revision as described in Section 3.3.9.2 . If one of the checks fails, return error code 0x07. <i>ComponentComparisonStamp</i> >= <i>ActiveComponentComparisonStamp</i> as reported by <i>GetFirmwareParameters</i> . If not, return error code 0x1- 0x2 accordingly.	Pass component information. Contains the component information table for a specific component including <i>ComponentClassificationIndex</i> , <i>ComponentClassification</i> , and version details. Number of commands is the number of components to update. Firmware should allow downgrade only up to security version.	10.8.4.2.4
UpdateComponent		Issue erase command to NVM. (Address in the NVM of the 1st bank pointer of the relevant bank). Respond success or other error code.	Request updating a specific firmware component. See Section 3.3.10.2 .	10.8.4.2.5
	RequestFirmwareData	Store internally the received data until 4 KB received, then issue an update NVM command. Respond success or other error code.	See Section 3.3.10.3 .	10.8.4.1.2
	TransferComplete	Verify the image authentication. The following is according to Section 3.3.5.6 (OROM)/ Section 3.3.5.6.2 (NVM), Step 6-Step 8 .		10.8.4.1.3
	VerifyComplete	Validation checks same as done by NVM Update admin command. Verify that <i>ComponentVersionString</i> , <i>ComponentComparisonStamp</i> , <i>ComponentSetVersionString</i> as defined in (Section 10.8.4.1.1) is passed before the download matches the downloaded NVM/ OROM bank, see in Section 10.8.5.2.1 .		10.8.4.1.4
	ApplyComplete			10.8.4.1.5

Table 10-70. Action Upon Reception of Commands - NVM Section [continued]

BMC Command	Firmware Device Command	Firmware Device Action	Remarks	Section Reference
ActivateFirmware		Update PFA according to the TLVs in <i>PackageData</i> . Respond success or other error code.	See Section 3.3.5.2.1 – Results in validate the new bank and invalidate the old bank. See Section 10.11.2.7.5 .	10.8.4.2.6

10.8.5.4 PLDM Events and Commands

PLDM events are to be answered by firmware during NVM update.

PLDM events have higher priority over PLDM update NVM commands.

PLDM commands are supported regardless of PLDM FWU enable bit in NVM.

PLDM Bit	PLDM FWU Bit	PLDM Commands	PLDM FWU Commands	Legal
Enabled	Enabled	Supported	Supported	Yes
Enabled	Disabled	Supported	Not supported	Yes
Disabled	Disabled	Drop	Drop	Yes
Disabled	Enabled	Drop	Drop	No

10.8.5.5 Reset During Update

The firmware should return to IDLE mode only in POR and EMPR. This is the case now anyway, so no additional handling is required.

In all other reset cases, the firmware update continues the update.

10.8.5.6 Activation Methods

The activation and reset after update is done by BMC.

The firmware *ApplyComplete* command includes the type of reset required.

Option ROM update requires PERST to take effect.

10.8.6 RDE Support

RDE is a protocol defined to allow exposure of devices directly to the Redfish infrastructure.

System-level architecture for RDE is presented in [Figure 10-15](#). The Administrator issues Redfish commands over HTTP/HTTPS. The Management Controller (MC), which supports Platform Level Data Model (PLDM) and RDE protocols, converts Redfish commands to the binary form used by the RDE Device. The RDE-capable device responds to MC over PLDM using BEJ format.

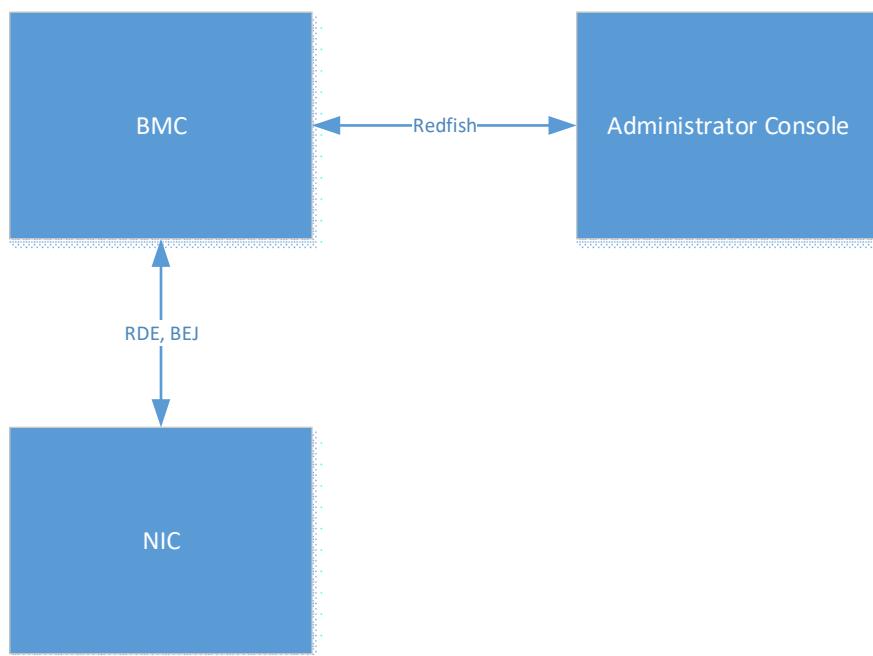


Figure 10-15. RDE Setup

Note: RDE is one of the tools that can change the adapter non-volatile configuration. As there are already software tools in place managing that configuration (for example, UEFI driver), conflicts might arise between these settings. The different software and firmware tools do not attempt to mitigate these conflicts. The system integrator is expected to make sure that configuration commands (sent via whichever API) do not conflict.

10.8.6.1 Implementation Guidelines

10.8.6.1.1 Presence of Payload in Command/Response

This section defines which actions expect the *contains_request_payload* flag to be set in the RDEOperationInit command, which sets the *HaveResultPayload* flag in the response, and which expects a valid *OperationLocator* field as part of the command.

Field	Read	Update	Replace	Create	Delete	Action	Head
contains_request_payload	MUST NOT	MUST	MUST	CAN	MUST NOT	MUST NOT	MUST NOT
HaveResultPayload	MUST	MUST NOT	MUST NOT	MUST NOT	MUST NOT	MUST NOT	MUST NOT
OperationLocator	CAN	CAN	CAN	CAN	CAN	MUST NOT	CAN

10.8.6.1.2 UPDATE, REPLACE Operation Guidelines

If an operation of type UPDATE or REPLACE is received with a BEJ payload containing a mix of Unknown properties (properties with sequence numbers that do not match the ones in the firmware's dictionary), Read-Only properties, and Read-Write properties, the operation silently ignores all Unknown and Read-Only properties and behaves as if the payload contains Read-Write properties only.

If the payload of a write operation contains Unknown and Read-Only properties exclusively, the operation ignores the entire payload, performs no write, and the operation fails with ERROR_UNSUPPORTED.

If no payload is provided or is provided but does not contain any properties at all, the operation fails with ERROR_INVALID_DATA.

10.8.6.1.3 String Handling

Any string received with a null character in the middle of the string is treated as an invalid input.

10.8.6.1.4 Resource Permissions

Each resource has associated permissions, which determine the types of operations permitted to be performed on that resource. If an RDE operation is invoked with an Operation Type forbidden by that resource, operation initialization fails with a return code of ERROR_NOT_ALLOWED. See [Table 10-74, "Resources Parameters"](#) for a list of allowed operation per resource. Permissions on a Property level are not individually supported, and all errors are silently ignored.

10.8.6.1.5 Resources Exposing @Redfish.Settings Attr.

For the *NetworkDeviceFunction*, *EthernetInterface*, *Port*, and *NetworkAdapter* schemas, there are two sets of resources, as follows:

Schema	Regular Resource ID (RO)	Settings Resource ID (RW)
NetworkDeviceFunction	200-207	210-217
EthernetInterface	400-407	410-417
Port	100-107	110-117
NetworkAdapter	1	11

Into each regular instance of the above resources, the following is inserted:

```
"@Redfish.Settings" : {
  "@odata.type": "#Settings.v1_3_0.Settings",
  "SupportedApplyTimes": ["OnReset"],
  "SettingsObject": {
    "@odata.id": 112    // bejResourceLink to Settings Resource NetworkPort 3
  }
}
```

When a write to one of these resources needs to be done, the Settings Resource (11, 110-117, 210-217, 410-417) is accessed. The Regular Resource and all its properties are RO. All changes written to by a Settings Resource must delay their application until at reset. After successful RDE Operation completion, the pending setting is stored in non-volatile memory. Any pending changes to the Settings Resources are applied at next PCI reset or EMP reset (internal firmware reset).

10.8.6.2 RDE Commands Summary

All the RDE commands should be implemented according to DSP0218 and follow the limitations/guidelines as defined in this chapter.

Table 10-71. RDE Commands Summary Table

Command	Command Code	Command Requirement for RDE Device	Section Reference
Discovery and Schema Management Commands			
NegotiateRedfishParameters	0x01	Supported	10.8.6.3.1
NegotiateMediumParameters	0x02	Supported	10.8.6.3.2
GetSchemaDictionary	0x03	Supported	10.8.6.3.3
GetSchemaURI	0x04	Supported	10.8.6.3.4
GetResourceETag	0x05	Supported	10.8.6.3.5
GetOEMCount	0x06	Supported	10.8.6.3.6
GetOEMName	0x07	Supported	10.8.6.3.7
GetRegistryCount	0x08	Supported	10.8.6.3.8
GetRegistryDetails	0x09	Supported	10.8.6.3.9
SelectRegistryVersion	0x0A	Not Supported	---
GetMessageRegistry	0x0B	Supported	10.8.6.3.10
GetSchemaFile	0x0C	Not Supported	---
Reserved	0x0D - 0x0F	---	---
RDE Operation and Task Commands			
RDEOperationInit	0x10	Supported	10.8.6.3.11
SupplyCustomRequestParameters	0x11	Supported	10.8.6.3.12
RetrieveCustomResponseParameters	0x12	Not Supported	10.8.6.3.13
RDEOperationComplete	0x13	Supported	10.8.6.3.14
RDEOperationStatus	0x14	Supported	10.8.6.3.15
RDEOperationKill	0x15	Supported	10.8.6.3.16
RDEOperationEnumerate	0x16	Supported	10.8.6.3.17
Reserved	0x17 - 0x2F	---	---
Multi-part Transfer Commands			
RDEMultipartSend	0x30	Supported	10.8.6.3.18
RDEMultipartReceive	0x31	Supported	10.8.6.3.19
Reserved	0x32 - 0x3F	---	---
Reserved			
Reserved	0x40 - 0xFF	---	---

10.8.6.3 Command Details

10.8.6.3.1 NegotiateRedfishParameters (0x01)

Type	Request Data
uint8	MCConcurrencySupport Firmware checks that this field is bigger than zero and ignores the value, as the device exposes single concurrent command support. If <i>MCConcurrencySupport</i> is equal to zero, return a completion code of <i>ERROR_INVALID_DATA</i> .
bitfield16	MCFeatureSupport [15:9] = reserved [8] = BEJ v1.1 encoding and decoding supported; 1b = yes [7] = events_supported — 1b = yes [6] = action_supported — 1b = yes [5] = replace_supported — 1b = yes [4] = update_supported — 1b = yes [3] = delete_supported — 1b = yes [2] = create_supported — 1b = yes [1] = read_supported — 1b = yes. [0] = head_supported — 1b = yes Firmware checks that Bit 8 is set before using <i>bejRegistryItem</i> .
Type	Response Data
enum8	CompletionCode Value: {PLDM_BASE_CODES}
uint8	DeviceConcurrencySupport Return 1.
bitfield8	DeviceCapabilitiesFlags Capabilities for this RDE device. [7:3] = Reserved [2] = BEJ v1.1 encoding and decoding supported — 1b = yes [1] = expand_support — 0b = No support [0] = atomic_resource_read — 0b = No support
bitfield16	DeviceFeatureSupport Operations and functionality supported by this RDE device. For each, 1b indicates supported and 0b indicates not supported. [15:8] = Reserved [7] = events_supported — 1b = yes [6] = action_supported — 1b = yes [5] = replace_supported — 1b = yes [4] = update_supported — 1b = yes [3] = delete_supported — 0b = no [2] = create_supported — 0b = no [1] = read_supported — 1b = yes [0] = head_supported — 1b = yes
uint32	DeviceConfigurationSignature A signature calculated across all RDE PDRs and dictionaries that the RDE device supports. Use OCS HCU engine with <i>ALGORITHM_MODE</i> = <i>SHA256</i> . Take lower 32 bit. Each PDR and dictionary is padded to 512 bits before added to the calculation.
varstring	DeviceProviderName An informal name for the RDE device. If a VPD Identifier String (tag 0x82) exists, use it as <i>DeviceProviderName</i> . Otherwise, use "Name" property at <i>NetworkInterface</i> .

10.8.6.3.2 NegotiateMediumParameters (0x02)

Type	Request Data
uint32	MCMaximumTransferChunkSizeBytes An indication of the maximum amount of data the MC can support for a single message transfer. A value of less than 64 bytes is considered as an error with ERROR_INVALID_DATA completion code, as it does not support minimal MCTP packet size of 64 bytes.
Type	Response Data
enum8	CompletionCode Value: {PLDM_BASE_CODES}
uint32	DeviceMaximumTransferChunkSizeBytes Report 2048 bytes.

After reception of this command, the device uses a maximal transfer chunk of Min(DeviceMaximumTransferChunkSizeBytes, MCMaximumTransferChunkSizeBytes) for MultipartSend/Receive commands.

The default value before reception of this command is 64 bytes (one message per MCTP packet).

If received on multiple media, use the minimum of all media for MultipartReceive response and accept MultipartSend with the maximum of all media values.

10.8.6.3.3 GetSchemaDictionary (0x03)

Type	Request Data
uint32	ResourceID The <i>ResourceID</i> of any resource in the Redfish Resource PDR from which to retrieve the dictionary. See Table 10-72 for supported Resource IDs. If resource is not supported, return a completion code of ERROR_NO_SUCH_RESOURCE.
schemaClass	RequestedSchemaClass The class of schema being requested. See Table 10-72 for supported schema per Resource IDs. If the a schema of the type requested is not supported, return a completion code of ERROR_UNSUPPORTED unless the supplied Resource ID does not correspond to a collection, but the <i>RequestedSchemaClass</i> is COLLECTION_MEMBER_TYPE, in which case, return an ERROR_INVALID_DATA completion code.
Type	Response Data
enum8	CompletionCode Value: { PLDM_BASE_CODES, ERROR_UNSUPPORTED, ERROR_NO_SUCH_RESOURCE }
uint8	DictionaryFormat Return 0x00.
uint32	TransferHandle In conjunction with a non-failed <i>CompletionCode</i> , the RDE device returns a valid transfer handle as described in Table 10-72 .

Table 10-72. Handle per Resource

Schema (*.json)	Dictionary Class Supported	Resource ID
Event	EVENT	0xFFFF FFFF or any of the other Resource IDs below.
redfish-payload-annotations	ANNOTATION	
redfish-error	ERROR	
Registry	REGISTRY	0xFFFF FFFF
NetworkAdapter	MAJOR	1 11
PCIeDevice	MAJOR	3
NetworkInterface	MAJOR	5
PortCollection	MAJOR	10
	COLLECTION_MEMBER_TYPE	10
PCIeFunctionCollection	MAJOR	30
NetworkDeviceFunctionCollection	MAJOR	20
NetworkDeviceFunction	COLLECTION_MEMBER_TYPE	20
Port	MAJOR	100-107 110-117
NetworkDeviceFunction	MAJOR	200-207 210-217
PCIFunction	MAJOR	300-307
EthernetInterface	MAJOR	400-407 410-417
EthernetInterfaceCollection	MAJOR	40
	COLLECTION_MEMBER_TYPE	40
NetworkAdapterMetrics	MAJOR	600
PortMetrics	MAJOR	700-707
NetworkDeviceFunctionMetrics	MAJOR	800-807

10.8.6.3.4 GetSchemaURI (0x04)

Type	Request Data
uint32	ResourceID The <i>ResourceID</i> of any resource in the Redfish Resource PDR from which to retrieve the URI. See Table 10-73 for supported Resource IDs. If resource is not supported, return a completion code of ERROR_NO_SUCH_RESOURCE.
schemaClass	RequestedSchemaClass The class of schema being requested. If the a schema of the type requested is not supported, return a completion code of ERROR_UNSUPPORTED.
uint8	OEMExtensionNumber No OEM extensions. Return a completion code of ERROR_INVALID_DATA for any non-zero value.

Type	Response Data
enum8	CompletionCode Value: <pre>{ PLDM_BASE_CODES, ERROR_UNSUPPORTED, ERROR_NO_SUCH_RESOURCE }</pre>
uint8	StringFragmentCount Return 1
varstring	SchemaURI [0] URI string fragment for the schema. The reassembled string is the canonical URI for the JSON Schema used by the RDE Device.

Table 10-73. URI per Resource

Schema	Class	Resource ID	URI
All schemas	EVENT	0xFFFF FFFF or any of the other Resource IDs below.	https://redfish.dmtf.org/schemas/Event.json
	ANNOTATION		https://redfish.dmtf.org/schemas/redfish-payload-annotations.v1_0_2.json
	ERROR		https://redfish.dmtf.org/schemas/v1/redfish-error.v1_0_0.json
NetworkAdapter	MAJOR	1 11	https://redfish.dmtf.org/schemas/NetworkAdapter.v1_7_0.json
NetworkInterface	MAJOR	5	https://redfish.dmtf.org/schemas/NetworkInterface.v1_2_0.json
PCIeDevice	MAJOR	3	https://redfish.dmtf.org/schemas/PCIeDevice.v1_4_0.json (not implemented)
PortCollection	COLLECTION_MEMBER_TYPE	10	https://redfish.dmtf.org/schemas/Port.json
	MAJOR		https://redfish.dmtf.org/schemas/PortCollection.json
NetworkDeviceFunctionCollection	COLLECTION_MEMBER_TYPE	20	https://redfish.dmtf.org/schemas/NetworkDeviceFunction.json
	MAJOR		https://redfish.dmtf.org/schemas/NetworkDeviceFunctionCollection.json
PCIeFunctionCollection	COLLECTION_MEMBER_TYPE	30	https://redfish.dmtf.org/schemas/PCIeFunction.json
	MAJOR		https://redfish.dmtf.org/schemas/PCIeFunctionCollection.json
Port	MAJOR	100-107 110-117	https://redfish.dmtf.org/schemas/NetworkPort.v1_6_0.json
NetworkDeviceFunction	MAJOR	200-207 210-217	https://redfish.dmtf.org/schemas/NetworkDeviceFunction.v1_6_0.json
PCIFunction	MAJOR	300-307	https://redfish.dmtf.org/schemas/PCIeFunction.v1_2_3.json
EthernetInterface	MAJOR	400-407 410-417	https://redfish.dmtf.org/schemas/EthernetInterface.v1_5_1.json
EthernetInterfaceCollection	COLLECTION_MEMBER_TYPE	40	https://redfish.dmtf.org/schemas/EthernetInterface.json
	MAJOR		https://redfish.dmtf.org/schemas/EthernetInterfaceCollection.json

Table 10-73. URI per Resource [continued]

Schema	Class	Resource ID	URI
NetworkAdapterMetrics	MAJOR	600	http://redfish.dmtf.org/schemas/v1/NetworkAdapterMetrics.v1_0_0.json
PortMetrics	MAJOR	700-707	http://redfish.dmtf.org/schemas/v1/PortMetrics.v1_2_0.json
NetworkDeviceFunctionMetrics	MAJOR	800-807	http://redfish.dmtf.org/schemas/v1/NetworkDeviceFunctionMetrics.v1_1_0.json

10.8.6.3.5 GetResourceETag (0x05)

Type	Request Data
uint32	ResourceID The <i>ResourceID</i> of a resource in the Redfish Resource PDR for the instance from which to get an ETag digest, or 0xFFFF FFFF to get a global digest of all resource-based data within the RDE device. The supported <i>ResourceIDs</i> are listed in Table 10-73 . For non-existent <i>ResourceID</i> , ERROR_NO_SUCH_RESOURCE <i>CompletionCode</i> is returned
Type	Response Data
enum8	CompletionCode Value: { PLDM_BASE_CODES, ERROR_NO_SUCH_RESOURCE }
varstring	ETag The ETag string data; the string text format is UTF-8. ETag is calculated as lower 32 bits of SHA256 computed from content using OCS-HCU engine with ALGORITHM_MODE = SHA256. This field is omitted if the <i>CompletionCode</i> is not SUCCESS. Firmware should calculate the data that would have been returned for the READ operation of the <i>ResourceID</i> and calculate the ETag for it. It should be the same <i>ETAG</i> value as returned as part of a read completion of <i>ResourceID</i> . Use the same algorithm as for signature at GetSchemaDictionary. ETag is calculated only on the data that is immediately contained within the resource.

10.8.6.3.6 GetOEMCount (0x06)

This command enables the MC to retrieve the number of OEM extensions for a schema.

Type	Request Data
uint32	ResourceID The <i>ResourceID</i> of the resource in the Redfish Resource PDR from which to retrieve the OEM count. A <i>ResourceID</i> of 0xFFFF FFFF can be supplied to retrieve OEM counts for schemas common to all RDE Device resources (such as the event dictionary) without referring to an individual resource.
schemaClass	RequestedSchemaClass The class of schema being requested. Note: Redfish does not allow OEM extensions to Annotation and Registry schemas.

Type	Response Data
enum8	CompletionCode Value: { PLDM_BASE_CODES, ERROR_NO_SUCH_RESOURCE }
uint8	OEMCount The number of OEM extensions associated with the schema. For schema classes that do not support OEM extensions this value must be zero.

For a non-SUCCESS *CompletionCode*, only the *CompletionCode* field of the Response Data is returned.

10.8.6.3.7 GetOEMName (0x07)

This command enables the MC to retrieve information about the name associated with an OEM extension to a schema (including schemas for which OEM information is available in a Redfish Resource PDR). The firmware counts OEM extensions in lexicographic order.

Type	Request Data
uint32	ResourceID The <i>ResourceID</i> of any resource in the Redfish Resource PDR from which to retrieve an OEM name. A <i>ResourceID</i> of 0xFFFF FFFF can be supplied to retrieve OEM names for extensions to schemas common to all RDE Device resources (such as the event dictionary) without referring to an individual resource.
schemaClass	RequestedSchemaClass The class of schema being requested.
uint8	OEMIndex The zero-based index of the OEM extension about which information is to be retrieved. The total number of OEM extensions supported by an RDE Device for a given schema can be retrieved via the GetOEMCount command; the index supplied here should be less than that count.
Type	Response Data
enum8	CompletionCode Value: { PLDM_BASE_CODES, ERROR_NO_SUCH_RESOURCE } A response code of ERROR_INVALID_DATA is used to indicate when the supplied index does not exist in the schema or when the schema class does not support OEM schemas.
varstring	OEMName The OEM name associated with the extension.

For a non-SUCCESS *CompletionCode*, only the *CompletionCode* field of the Response Data is returned.

10.8.6.3.8 GetRegistryCount (0x08)

Type	Request Data
---	None.
Type	Response Data
enum8	CompletionCode Value: {PLDM_BASE_CODES}
uint8	RegistryCount The number of registries supported by the Device.

For a non-SUCCESS *CompletionCode*, only the *CompletionCode* field of the Response Data is returned.

10.8.6.3.9 GetRegistryDetails (0x09)

Type	Request Data
uint8	RegistryIndex The zero-based index of the message registry about which information is to be retrieved. The total number of registries supported by an RDE Device can be retrieved via the GetRegistryCount command; the index supplied here should not exceed that count.
Type	Response Data
enum8	CompletionCode Value: {PLDM_BASE_CODES} A response code of ERROR_INVALID_DATA is used when the supplied index does not correspond to a supported registry.
varstring	RegistryPrefix The Redfish prefix (name without version information) associated with the registry.
varstring	RegistryURI URI at which the registry schema is published.
enum8[2]	RegistryLanguage Language in which the registry is published, as an ISO 639-1 two-letter code.
enum8	VersionCount The number N of registry versions the RDE Device supports for this registry.
ver32	Version [0] First (newest) version of the registry supported.
...	...
ver32	Version [N - 1] Last (oldest) version of the registry supported.

For a non-SUCCESS *CompletionCode*, only the *CompletionCode* field of the Response Data is returned.

The firmware counts message registries in lexicographic order and returns message registry versions in reverse numeric order (most recent versions listed first). The RDE Device truncates the list and decreases the count as needed to ensure that the response message fits within the negotiated message size, thereby omitting mention of support for older versions.

Note: Currently, the lexicographic reverse numeric order is not an issue as only a single registry is supported.

10.8.6.3.10 GetMessageRegistry (0x0B)

This command enables the MC to retrieve the formal JSON registry for a Redfish message registry supported by the RDE device. After invoking the GetMessageRegistry command, the MC, upon receipt of a successful completion code and a valid read transfer handle, invokes one or more RDEMultiPartReceive commands to transfer data for the registry from the RDE Device. The MC can only have one dictionary, schema, or message registry retrieval in process from a given RDE Device at any time. In the event that the MC begins a dictionary, schema, or message registry retrieval when a previous retrieval has not yet completed (that is, more chunks of dictionary or schema data remain to be retrieved), the previous retrieval is implicitly aborted and the RDE Device discards any data associated with the transfer.

Type	Request Data
uint8	RegistryIndex The zero-based index of the message registry to be retrieved. The total number of registries supported by an RDE Device can be retrieved via the GetRegistryCount command; the index supplied here should not exceed that count.
Type	Response Data
enum8	CompletionCode Value: {PLDM_BASE_CODES} A response code of ERROR_INVALID_DATA is used when the supplied index does not correspond to a supported registry.
uint8	SchemaFormat Bitwise OR of two values: Text format: {RAW_UTF8 = 0; GZIP_UTF8 = 1} Schema format: {JSON = 0x10; CSDL = 0x20; YAML = 0x30} Taken from RDE Dictionaries NVM section.
uint32	TransferHandle A data transfer handle that the MC uses to retrieve the registry data via one or more RDEMultiPartReceive commands. In conjunction with a non-failed CompletionCode, the RDE Device returns a valid transfer handle.

For a non-SUCCESS CompletionCode, only the CompletionCode field of the Response Data is returned.

10.8.6.3.11 RDEOperationInit (0x10)

Type	Request Data
uint32	ResourceID The <i>ResourceID</i> of a resource in the Redfish Resource PDR for the data that is the target of this operation. The supported <i>ResourceIDs</i> are listed in Table 10-74 .
rdeOpID	OperationID Identification number for this Operation. Must be matched by all commands relating to this Operation.
enum8	OperationType The type of Redfish Operation being performed. The supported operation types per ResourceID are listed in Table 10-74 .
bitfield8	OperationFlags [7:3] = Reserved [2] = contains_custom_request_parameters — If 1b, the RDE Device should expect to receive a SupplyCustomRequestParameters command request before it may trigger the Operation [1] = contains_request_payload — If 0b, the Operation does not require data to be sent. This bit is expected to be set only if OperationType is OPERATION_UPDATE or OPERATION_REPLACE. [0] = locator_valid — If 0b, the Locator in the <i>OperationLocator</i> field is ignored
uint32	SendDataTransferHandle Handle to be used with the first MultipartSend command transferring BEJ formatted data for the operation. If no data is to be sent for this operation, or if the request payload fits entirely within this request message, it is 0x00000000 (see the <i>RequestPayloadLength</i> and <i>RequestPayload</i> fields below).
uint8	OperationLocatorLength Length in bytes of the <i>OperationLocator</i> for this Operation. This field is zero if the <i>locator_valid</i> bit in the <i>OperationFlags</i> field above is set to 0b.
uint32	RequestPayloadLength Length in bytes of the request payload in this message. This value is zero under either of the following conditions: - There is no request payload as indicated by <i>contains_request_payload</i> bit of the <i>OperationFlags</i> parameter above. - The entire payload cannot fit within this message, subject to the maximum transfer chunk size as determined at registration time via the NegotiateMediumParameters command.
bejLocator	OperationLocator BEJ Locator indicating where the new Operation is to take place within the resource specified in <i>ResourceID</i> . Supported for Read, Update, and Action operations. This field is omitted if the <i>OperationLocatorLength</i> field above is set to zero.
null or bejEncoding	RequestPayload The request payload. The format of this parameter is null (consisting of zero bytes) if the <i>RequestPayloadLength</i> above is zero. It is bejEncoding otherwise.
Type	Response Data
enum8	CompletionCode Return ERROR_NOT_READY if firmware is in the middle of an operation preventing the access. For example: - Configuration change - Reset - Firmware update Return ERROR_NOT_ALLOWED if operation not allowed for the requested resource (e.g. Update a RO resource). Return ERROR_NO_SUCH_RESOURCE if resource ID is not advertised by the device. Return ERROR_CANNOT_CREATE_OPERATION if attempt to create a task while another task is in-flight. Return ERROR_OPERATION_EXISTS if attempt to re-create the same action with same {ResourceID, OperationID}. Return ERROR_INVALID_DATA if OperationID MSB bit is cleared (device owned operation ID).

enum8	OperationStatus Return OPERATION_NEEDS_INPUT if <i>OperationFlags.contains_custom_request_parameters</i> =1 OR <i>OperationFlags.contains_request_payload</i> =1 AND <i>SendDataTransferHandle</i> <> 0 Return OPERATION_RUNNING if no input data needed and operation did not complete within T1. Return OPERATION_HAVE_RESULTS if no input data needed and operation completed within T1, and there are parameters to return that do not fit within the response. Return OPERATION_COMPLETED if no input data needed and operation completed within T1, and there are parameters to return that fit within the response or there are no parameters to return. Return OPERATION_FAILED in case of error.
uint8	CompletionPercentage Return 255 if operation is not valid. Return zero if the Operation has not yet been triggered or if the Operation has failed. Return 254 otherwise.
uint32	CompletionTimeSeconds Return 0xFFFF FFFF - no support
bitfield8	OperationExecutionFlags [7:4] = Reserved [3] = CacheAllowed — 1b = yes; 0b = no Set to 0b for Operations other than read, head. Set to 0b unless Operation has finished. See Section 10.8.6.5.2 for list cacheable parameters. A value of 1b is reported if all the read parameters are cacheable. [2] = HaveResultPayload — 1b = yes. 0b = no. Set to 0b if Operation has not finished. Set to 1b for completed Read operations This bit is set only if <i>OperationType</i> is OPERATION_GET. [1] = HaveCustomResponseParameters — 1b = yes. 0b = no Set to 0b if Operation has not finished or no custom response parameters available even if operation is finished. Set to 1b for all other cases. Currently not set. [0] = TaskSpawned — 1b = yes. 0b = no. Set to 1 for tasks longer than PT1 - Request-to-response time as specified at DSP0240. Firmware can start the task and start the timer and return the response depending if it succeeded to finish within PT1 or not, if not it will be a long running task.
uint32	ResultTransferHandle 0x00000000: No data to transfer from endpoint to MC. 0xFFFFFFFF: Operation not complete. Other: Handle provided by firmware - should not be a direct address in memory.
bitfield8	PermissionFlags Indicates the access level granted to the resource targeted by the Operation. Is set to 0x00 by the RDE device and ignored by the MC if the completion code is not ERROR_NOT_ALLOWED and <i>OperationType</i> is neither OPERATION_HEAD nor OPERATION_READ. [7: 6] = Reserved for future use [5] = Head access — 1b = Head read allowed [4] = Delete access — 1b = Delete allowed (currently not supported by the E610) [3] = Create access — 1b = Create allowed (currently not supported by the E610) [2] = Replace access — 1b = Replace allowed; 0b = replace not allowed [1] = Update access — 1b = Update allowed; 0b = update not allowed [0] = Read access — 1b = Read allowed
uint32	ResponsePayloadLength As described in specification. Currently no <i>@Message.ExtendedInfo</i> planned in case of error.
varstring	ETag String data for an ETag digest of the target resource; the string text format is UTF-8. The ETag is skipped (an empty string returned in this field) for any of the following actions: Action, Delete, Replace, and Update. The ETag is also skipped (an empty string returned in this field) if execution of the Operation has failed or not yet finished. To create ETag, use OCS HCU engine with ALGORITHM_MODE = SHA256 and take lower 32 bits of the result.
null or bejEncoding	ResponsePayload The response payload. The format of this parameter shall be null (consisting of zero bytes) if the <i>ResponsePayloadLength</i> above is zero. It is bejEncoding otherwise.

Note: Firmware should save the {*ResourceID*, *OperationID*}, context until *RDEOperationComplete*., as it is used by MC as a handle for all intermediate commands. It should also keep the *SendDataTransferHandle* and compare it to the *DataTransferHandle* received in the *MultipartSend* command. It should also keep the returned *ResultTransferHandle* and compare it with the value provided in the *MultipartReceive*.

Note: A payload (either in the command or in the separate data transfer) is expected only for update and replace operations.

Note: When bit [1] in *OperationFlags* field is not set, the *PayloadLength* and *RequestTransferHandle* fields must be set to zero. Otherwise, *ERROR_CANNOT_CREATE_OPERATION* error response is returned.

Table 10-74. Resources Parameters

Schema	Resource ID	Allowed Operations
NetworkAdapter	1	Read, Head, Update, Replace, Action
	11	Read, Head, Update, Replace
NetworkInterface	5	Read, Head, Update, Replace
PCIeDevice	3	Read, Head, Update, Replace (not implemented)
PortCollection	10	Read, Head
NetworkDeviceFunctionCollection	20	Read, Head
Port	100-107	Read, Head, Action, Update, Replace
	110-117	Read, Head, Update, Replace
NetworkDeviceFunction	200-207	Read, Head, Update, Replace
PCIFunction	300-307	Read, Head, Update, Replace
EthernetInterface	400-407	Read, Head, Update, Replace
EthernetInterfaceCollection	40	Read, Head
Setting Resources	210-217	Read, Head, Update, Replace
	410-417	Read, Head, Update, Replace
NetworkAdapterMetrics	600	Read, Head
PortMetrics	700-707	Read, Head
NetworkDeviceFunctionMetrics	800-807	Read, Head

10.8.6.3.12 SupplyCustomRequestParameters (0x11)

Type	Request Data
uint32	ResourceID The <i>ResourceID</i> of a resource in the Redfish Resource PDR for the data that is the target of this operation. The supported <i>ResourceIDs</i> are listed in Table 10-74 .
rdeOpID	OperationID The { <i>ResourceID</i> , <i>OperationID</i> } should match the value set in the RDEOperationInit command. Otherwise, an ERROR_UNEXPECTED error is returned.
uint16	LinkExpand This value is ignored.
uint16	CollectionSkip Defines the number of elements to skip in a GET response. Ignored for other actions. Zero means return all.
uint16	CollectionTop Defines the maximum number of elements to return in a GET response. Ignored for other actions. 0xFFFF means return all.
uint16	PaginationOffset This value is ignored. The device does not do pagination.
enum8	ETagOperation If <i>ETagOperation</i> = ETAG_IF_MATCH = 1, do the action only if the calculated ETag for the resource == ETag [0]. If <i>ETagOperation</i> = ETAG_IF_MATCH and EtagCount <> 1, return a ERROR_INVALID_DATA error. If <i>ETagOperation</i> = ETAG_IF_NONE_MATCH = 2, do the action only if the calculated ETag for the resource <> all of {ETag [0] ... ETag[ETagCount-1]. If <i>ETagOperation</i> = ETAG_IF_NONE_MATCH and EtagCount == 0, return a ERROR_INVALID_DATA error. If <i>ETagOperation</i> = ETAG_IGNORE = 0 and EtagCount <> 0 return a ERROR_INVALID_DATA error. If <i>ETagOperation</i> >= 3, return a ERROR_INVALID_DATA error.
uint8	ETagCount Number of ETags supplied in this message. Should be zero if <i>ETagOperation</i> above is ETAG_IGNORE, and nonzero otherwise.
varstring	ETag [0] String data for first ETag, if <i>ETagCount</i> > 0. This string is UTF-8 format. ETag is calculated as lower 32 bits of SHA256 computed from content using OCS-HCU engine.
...	Additional ETags
uint8	HeaderCount The number of custom headers being supplied in this operation. Currently no support, so must be zero.
varstring	HeaderName [0] The name of the header, including the X- prefix
varstring	HeaderParameter [0] The parameter or parameters associated with the header. The MC can pre-process these (though any such preprocessing is outside the scope of this specification) or convey them exactly as received.
...	...

Type	Response Data
enum8	CompletionCode Values: <pre>{ PLDM_BASE_CODES, ERROR_OPERATION_ABANDONED, ERROR_OPERATION_FAILED, ERROR_UNSUPPORTED, ERROR_UNEXPECTED, ERROR_UNRECOGNIZED_CUSTOM_HEADER, ERROR_ETAG_MATCH, ERROR_NO_SUCH_RESOURCE }</pre> When an unknown <i>ResourceID</i> is supplied, ERROR_NO_SUCH_RESOURCE is returned. If <i>OperationID</i> references an unknown operation, or <i>OperationID</i> and <i>ResourceID</i> do not both match an existing operation, ERROR_UNEXPECTED is returned. If unsupported ETag operation value, return ERROR_UNSUPPORTED/ Response codes ERROR_UNSUPPORTED and ERROR_UNRECOGNIZED_CUSTOM_HEADER are used to indicate that an unsupported request parameter was sent. These responses represent an Operational failure, not a command failure.
enum8	OperationStatus Values: <pre>{ OPERATION_INACTIVE = 0, OPERATION_NEEDS_INPUT = 1, OPERATION_TRIGGERED = 2, OPERATION_RUNNING = 3, OPERATION_HAVE_RESULTS = 4, OPERATION_COMPLETED = 5, OPERATION_FAILED = 6, OPERATION_ABANDONED = 7 }</pre>
uint8	CompletionPercentage Return 255 if operation is not valid. Return zero if the Operation has not yet been triggered or if the Operation has failed. Return 254 otherwise.
uint32	CompletionTimeSeconds Return 0xFFFF FFFF.
bitfield8	OperationExecutionFlags See <i>OperationExecutionFlags</i> in RDEOperationInit (Section 10.8.6.3.11).
uint32	ResultTransferHandle See <i>ResultTransferHandle</i> in RDEOperationInit (Section 10.8.6.3.11).
bitfield8	PermissionFlags See <i>PermissionFlags</i> in RDEOperationInit (Section 10.8.6.3.11).
uint32	ResponsePayloadLength See <i>ResponsePayloadLength</i> in RDEOperationInit (Section 10.8.6.3.11).
varstring	ETag See <i>ETag</i> in RDEOperationInit (Section 10.8.6.3.11).
null or bejEncoding	ResponsePayload See <i>ResponsePayload</i> in RDEOperationInit (Section 10.8.6.3.11).

10.8.6.3.13 RetrieveCustomResponseParameters (0x12)

This command is currently not supported, as the NIC does not provide any Custom Response parameters.

Type	Request Data
uint32	ResourceID The <i>ResourceID</i> of a resource in the Redfish Resource PDR for the data that is the target of this operation. The supported <i>ResourceIDs</i> are listed in Table 10-74 .
rdeOpID	OperationID The { <i>ResourceID</i> , <i>OperationID</i> } should match the value set in the RDEOperationInit command. Otherwise, an ERROR_UNEXPECTED error is returned.
Type	Response Data
enum8	CompletionCode Values: { PLDM_BASE_CODES, ERROR_OPERATION_ABANDONED, ERROR_OPERATION_FAILED, ERROR_UNEXPECTED, ERROR_NO_SUCH_RESOURCE } When an unknown <i>ResourceID</i> is supplied, ERROR_NO_SUCH_RESOURCE is returned. If <i>OperationID</i> references an unknown operation, or <i>OperationID</i> and <i>ResourceID</i> do not both match an existing operation, ERROR_UNEXPECTED is returned.
uint32	DeferralTimeframe Return 0xFF (unknown).
uint32	NewResourceID Return 0 - no support for Create command.
uint8	ResponseHeaderCount Return 0 - no support for custom headers.
varstring	HeaderName [0] N/A
varstring	HeaderParameter [0] N/A
...	...

10.8.6.3.14 RDEOperationComplete (0x13)

Type	Request Data
uint32	ResourceID The <i>ResourceID</i> of a resource in the Redfish Resource PDR for the data that is the target of this operation. The supported <i>ResourceIDs</i> are listed in Table 10-74 .
rdeOpID	OperationID The { <i>ResourceID</i> , <i>OperationID</i> } should match the value set in the RDEOperationInit command. Otherwise, an ERROR_UNEXPECTED error is returned.
Type	Response Data
enum8	CompletionCode Values: { PLDM_BASE_CODES, ERROR_UNEXPECTED, ERROR_NO_SUCH_RESOURCE } When an unknown <i>ResourceID</i> is supplied, ERROR_NO_SUCH_RESOURCE is returned. If <i>OperationID</i> references an unknown operation, or <i>OperationID</i> and <i>ResourceID</i> do not both match an existing operation, ERROR_UNEXPECTED is returned.

10.8.6.3.15 RDEOperationStatus (0x14)

Returns status of the current task.

Type	Request Data
uint32	ResourceID The <i>ResourceID</i> of a resource in the Redfish Resource PDR for the data that is the target of this operation. The supported <i>ResourceIDs</i> are listed in Table 10-74 .
rdeOpID	OperationID The { <i>ResourceID</i> , <i>OperationID</i> } should match the value set in the RDEOperationInit command. Otherwise, an ERROR_UNEXPECTED error is returned.
Type	Response Data
enum8	CompletionCode Values: { PLDM_BASE_CODES, ERROR_UNSUPPORTED, ERROR_NO_SUCH_RESOURCE, ERROR_ETAG_MATCH, ERROR_UNRECOGNIZED_CUSTOM_HEADER } When an unknown <i>ResourceID</i> is supplied ERROR_NO_SUCH_RESOURCE is returned. If <i>OperationID</i> references an unknown operation, or <i>OperationID</i> and <i>ResourceID</i> do not both match an existing operation, ERROR_UNEXPECTED is returned. The completion code for RDEOperationStatus is one of the following: SUCCESS: A valid active RDE Operation was referenced in the <i>OperationID</i> request field and it is not in the failed state. The actual current status of the RDE Operation is returned in the <i>OperationStatus</i> field - OR - an inactive RDE Operation was referenced in the <i>OperationID</i> request field. <i>OperationStatus</i> is OPERATION_INACTIVE in this case. ERROR_UNSUPPORTED, ERROR_ETAG_MATCH, ERROR_UNRECOGNIZED_CUSTOM_HEADER: A valid active RDE Operation was referenced in the <i>OperationID</i> request field, but the Operation failed with the specified status code. <i>OperationStatus</i> is OPERATION_FAILED in this case. These responses indicate a failure in the RDE Operation, not a failure in the RDEOperationStatus command.

enum8	OperationStatus According to state machine
uint8	CompletionPercentage Return 255 if operation is not valid. Return zero if the Operation has not yet been triggered or if the Operation has failed. Return 254 otherwise.
uint32	CompletionTimeSeconds Return 0xFFFF FFFF.
bitfield8	OperationExecutionFlags See <i>OperationExecutionFlags</i> in RDEOperationInit (Section 10.8.6.3.11).
uint32	ResultTransferHandle See <i>ResultTransferHandle</i> in RDEOperationInit (Section 10.8.6.3.11).
bitfield8	PermissionFlags See <i>PermissionFlags</i> in RDEOperationInit (Section 10.8.6.3.11).
uint32	ResponsePayloadLength See <i>ResponsePayloadLength</i> in RDEOperationInit (Section 10.8.6.3.11).
varstring	ETag See <i>ETag</i> in RDEOperationInit (Section 10.8.6.3.11).
null or bejEncoding	ResponsePayload See <i>ResponsePayload</i> in RDEOperationInit (Section 10.8.6.3.11).

10.8.6.3.16 RDEOperationKill (0x15)

Type	Request Data
uint32	ResourceID The <i>ResourceID</i> of a resource in the Redfish Resource PDR for the data that is the target of this operation. The supported <i>ResourceIDs</i> are listed in Table 10-74, "Resources Parameters".
rdeOpID	OperationID The { <i>ResourceID</i> , <i>OperationID</i> } should match the value set in the RDEOperationInit command. Otherwise, an ERROR_UNEXPECTED is returned.
bitfield8	KillFlags Flags for killing the Operation: [7:3] = Reserved for future use [2] = discard_results — If 1b and the RDE Device is in the HAVE_RESULTS state for this operation, the results of the operation are discarded and the operation state set to Inactive. The MC must not set the discard_results bit in conjunction with any other bits in the killflags. In the event that the MC violates this restriction, the RDE Device responds with completion code ERROR_INVALID_DATA and stops processing the request. [1] = run_to_completion — If 1b, the Operation should be run to completion but no further response should be sent to the MC. The MC does not set the run_to_completion bit without also setting the discard_record bit. [0] = discard_record — If 1b and the kill command returns success, the RDE device discards internal records associated with this Operation as soon as it is killed; the RDE device should not expect the MC to call RedfishOperationComplete for this Operation. If the Operation has spawned a Task, the RDE device does not create an Event when execution is finished.

Type	Response Data
enum8	CompletionCode Value: <pre>{ PLDM_BASE_CODES, ERROR_OPERATION_ABANDONED, ERROR_OPERATION_FAILED, ERROR_OPERATION_UNKILLABLE, ERROR_NO_SUCH_RESOURCE, ERROR_UNEXPECTED }</pre> When an unknown <i>ResourceID</i> is supplied, ERROR_NO_SUCH_RESOURCE is returned. If <i>OperationID</i> references an unknown operation, or <i>OperationID</i> and <i>ResourceID</i> do not both match an existing operation, ERROR_UNEXPECTED is returned. If <i>KillFlags.run_to_completion</i> is set, but <i>KillFlags.discard_record</i> is cleared, return ERROR_INVALID_DATA. If both <i>KillFlags.run_to_completion</i> and <i>KillFlags.discard_record</i> are set, according to current state returns the following response: • If state is NEED_INPUT, return ERROR_UNEXPECTED. • If state is HAVE_RESULTS, return ERROR_OPERATION_UNKILLABLE/ • If state is FAILED, return ERROR_OPERATION_FAILED. • If state is ABANDONED, return ERROR_OPERATION_ABANDONED. • If state is other, return SUCCESS.

10.8.6.3.17 RDEOperationEnumerate (0x16)

Type	Request Data
N/A	This request contains no parameters.
Type	Response data.
enum8	CompletionCode Value: <pre>{PLDM_BASE_CODES}</pre>
unit16	OperationCount 0 if no active operation; 1 otherwise
unit32	ResourceID [0] The resource ID of the active action. Shall be omitted if OperationCount is zero
rdeOpID	OperationID [0] The Operation ID of the active action. Shall be omitted if OperationCount is zero
enum8	OperationType [0] The type of Operation. Shall be omitted if OperationCount is zero Values: <pre>{ OPERATION_HEAD = 0, OPERATION_READ = 1, OPERATION_UPDATE = 4, OPERATION_REPLACE = 5, OPERATION_ACTION = 6 }</pre> This field is omitted if <i>OperationCount</i> above is zero.

10.8.6.3.18 RDEMultiPartSend (0x30)

Type	Request Data
uint32	DataTransferHandle A handle to uniquely identify the chunk of data to be sent. If <i>TransferFlag</i> below is START or START_AND_END, this must match the <i>SendDataTransferHandle</i> that was supplied by the RDE Device in the response to RDEOperationInit. The <i>DataTransferHandle</i> supplied is either the initial handle to begin or restart a transfer, or the <i>NextDataTransferHandle</i> as specified in the previous chunk.
rdeOpID	OperationID The <i>OperationID</i> received in RDEOperationInit.
enum8	TransferFlag An indication of current progress within the transfer. The value START_AND_END indicates that the entire transfer consists of a single chunk. Values: <pre>{ START = 0, MIDDLE = 1, END = 2, START_AND_END = 3 }</pre>
uint32	NextDataTransferHandle The handle for the next chunk of data for this transfer; zero (0x00000000) if no further data. The firmware should keep this value internally until the next MultiPartSend.
uint32	DataLengthBytes The length in bytes N of data being sent in this chunk. This value and the Data bytes associated with it must not cause this request message to exceed the negotiated maximum transfer chunk size.
uint8	Data [0] The first byte of the current chunk of data. The <i>Data</i> field is omitted from the request message if the value of <i>DataLengthBytes</i> above is zero
...	...
uint8	Data [N-1] The last byte of the current chunk of data
uint32	DataIntegrityChecksum 32-bit CRC for the entirety of data (all parts concatenated together). Is omitted for non-final chunks (<i>TransferFlag</i> ≠ END or START_AND_END) in the transfer. For this specification, the CRC-32 algorithm with the polynomial: For this command, the CRC-32 algorithm with the polynomial $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ (same as the one used by IEEE 802.3) is used for the integrity checksum computation. The CRC computation involves processing a byte at a time with the least significant bit first.
Type	Response Data
enum8	CompletionCode Values: <pre>{ PLDM_BASE_CODES, ERROR_OPERATION_ABANDONED, ERROR_OPERATION_FAILED, ERROR_UNEXPECTED, ERROR_BAD_CHECKSUM }</pre> If <i>OperationID</i> is non-zero and references an unknown operation ERROR_UNEXPECTED is returned. If the <i>DataTransferHandle</i> does not correspond to a valid chunk, the RDE Device returns <i>CompletionCode</i> ERROR_INVALID_DATA.

enum8	TransferOperation The follow-up action that the RDE device is requesting of the MC: • XFER_FIRST_PART: Resend the initial chunk (restarting the transmission, such as if the checksum of data received did not match the DataIntegrityChecksum in the final chunk). • XFER_NEXT_PART: Send the next chunk of data. • XFER_ABORT: Stop the transmission and do not retry. In this case, the MC proceeds as if the transmission is permanently failed. • XFER_COMPLETE: no further follow-up needed, the transmission completed normally Values: <pre>{ XFER_FIRST_PART = 0, XFER_NEXT_PART = 1, XFER_ABORT = 2, XFER_COMPLETE = 3 }</pre>

10.8.6.3.19 RDEMultiPartReceive (0x31)

Type	Request Data
uint32	DataTransferHandle If <i>TransferOperation</i> below is XFER_FIRST_PART and the <i>OperationID</i> below is zero, this must match the <i>TransferHandle</i> supplied by the RDE Device in the response to the GetSchemaDictionary, GetMessageRegistry, and GetSchemaFile commands. If <i>TransferOperation</i> below is XFER_FIRST_PART and the <i>OperationID</i> below is nonzero, this must match the <i>SendDataTransferHandle</i> that was supplied by the RDE Device in the response to RDEOperationInit. If <i>TransferOperation</i> below is XFER_NEXT_PART, this must match the <i>NextDataHandle</i> supplied by the RDE Device with the previous chunk. The <i>DataTransferHandle</i> supplied is either the initial handle to begin or restart a transfer or the <i>NextDataTransferHandle</i> supplied with the previous chunk.
rdeOpID	OperationID The <i>OperationID</i> received in RDEOperationInit. A value of zero is used for Dictionary transfer (not part of an operation).
enum8	TransferOperation The portion of data requested for the transfer: • XFER_FIRST_PART: The MC is asking the transfer to begin or to restart from the beginning. • XFER_NEXT_PART: The MC is asking for the next portion of the transfer. • XFER_ABORT: The MC is requesting that the transfer be discarded. The RDE device may discard any internal data structures it is maintaining for the transfer. Values: <pre>{ XFER_FIRST_PART = 0, XFER_NEXT_PART = 1, XFER_ABORT = 2 }</pre>

Type	Response Data
enum8	CompletionCode Values: <pre>{ PLDM_BASE_CODES, ERROR_OPERATION_ABANDONED, ERROR_OPERATION_FAILED, ERROR_UNEXPECTED, ERROR_BAD_CHECKSUM }</pre> If <i>OperationID</i> is non-zero and references an unknown operation, ERROR_UNEXPECTED is returned. If the <i>DataTransferHandle</i> does not correspond to a valid chunk, the RDE Device returns <i>CompletionCode</i> ERROR_INVALID_DATA. If the transfer is aborted, the RDE device acknowledges this status by returning SUCCESS.
enum8	TransferFlag Values: <pre>{ START = 0, MIDDLE = 1, END = 2, START_AND_END = 3 }</pre> This field is omitted for a non-SUCCESS <i>CompletionCode</i> or if the transfer has been aborted.
uint32	NextDataTransferHandle The handle for the next chunk of data for this transfer; zero (0x00000000) if no further data. Calculated by firmware. This field is omitted for a non-SUCCESS <i>CompletionCode</i> or if the transfer has been aborted.
uint32	DataLengthBytes The length in bytes N of data being sent in this chunk. This value and the Data bytes associated with it must not cause this response message to exceed the negotiated maximum transfer chunk size. This field is omitted for a non-SUCCESS <i>CompletionCode</i> or if the transfer has been aborted.
uint8	Data [0] The first byte of current chunk of data. The <i>Data</i> field is omitted from the response message if the value of <i>DataLengthBytes</i> above is zero This field is omitted for a non-SUCCESS <i>CompletionCode</i> or if the transfer has been aborted.
...	...
uint8	Data [N-1] The last byte of the current chunk of data. This field is omitted for a non-SUCCESS <i>CompletionCode</i> or if the transfer has been aborted.
uint32	DataIntegrityChecksum 32-bit CRC for the entire block of data (all parts concatenated together). Is omitted for non-final chunks (TransferFlag ≠ END or START_AND_END) in the transfer or for aborted transfers. The recipient ignores this value except from the final transfer. For this command, the CRC-32 algorithm with the polynomial $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ (same as the one used by IEEE 802.3) is used for the integrity checksum computation. The CRC computation involves processing a byte at a time with the least significant bit first. Calculated by firmware.

10.8.6.4 State Machine

This section describes operating behavior of RDE support from RDE device-centric perspective. States that are a part of state-machine do not represent state of the RDE device but rather state for operation.

To present operation life cycle mechanisms, eight operational states for RDE device are introduced, as shown in [Table 10-75](#).

Table 10-75. RDE Device Operational States

State	Definition
INACTIVE	Default Operation state in which the RDE Device starts after initialization. Device is not processing an Operation, as it has not received an RDEOperationInit command from the MC.
NEED_INPUT	After receiving the RDEOperationInit command, the RDE Device moves to this state if it is expecting additional Operation-specific parameters or a payload that was not in-lined in the RDEOperationInit command.
TRIGGERED	Once the RDE Device receives everything it needs to execute an Operation, it begins executing it immediately.
TASK_RUNNING	If the RDE Device cannot complete the Operation within the time frame needed for the response to the command that triggered it, the RDE Device spawns a Task in which to execute the Operation asynchronously.
HAVE_RESULTS	When execution of the Operation produces a response parameters or a response payload that does not fit in the response message for the command that triggered the Operation (or detected its completion, if a Task was spawned or if there was a payload but no custom request parameters), the RDE Device remains in this state until the MC has collected all of these results.
COMPLETED	The RDE Device has completed processing of the Operation and awaits acknowledgment from the MC that it has received any Operation response data. This acknowledgment is done by the MC issuing the RDEOperationComplete command.
ABANDONED	If MC fails to progress the Operation through this state machine, the RDE Device may abort the Operation and mark it as abandoned.
FAILED	The MC has explicitly killed the Operation or an error prevented execution of the Operation.

Transitions between particular RDE operational states are shown in [Figure 10-16](#). State machine with all possible transitions is defined in *DSP0218: Platform Level Data Model (PLDM) for Redfish Device Enablement*, Section 9.2.3.2, Table 45.



The following schemas can be found in DSP8010 at <https://www.dmtf.org/standards/redfish>

Properties or options that are not mentioned here can be considered as "not supported".

1690

10.8.6.5.1 Resources Parameters

CacheAllowed - Relevant for RDEOperationInit for the read, head operations. Referring to RFC 7234, a value of "yes" is considered equivalent to Cache-Control response header value "public", and a value of "no" is considered equivalent to Cache-Control response header value "no-store". Typically, static data is allowed to be cached unless, for example, it represents sensitive data such as login credentials; data that changes over time is generally not marked as cacheable. A table in each schema defines if a parameter is writable and/or cacheable.

Note: To make it easier to port to other devices, it is recommended to keep the "E610" string as a variable in firmware.

Schemas that instantiate multiple resources are indexed by last two decimal digits of their *ResourceID*. The value of those two digits is Resource Offset, relevant for ID field.

10.8.6.5.2 Common Fields in All Schemas

The following parameters appear in many schemas and get the following values:

Property	Source	Cacheable	Writable	Notes
@odata.id	Returns bejResourceLink encoded resource ID.	Yes	No	
@odata.type	For simple schemas, a string built as: #Schema_name.vX_Y_Z.Schema_name For example, for NetworkInterface v1.1.1, the @odata_type is: #NetworkInterface.v1_1_1.NetworkInterface For collection schemas, a string built as: #Schema_name.Schema_name For example, for PortCollection, the @odata_type is: #PortCollection.PortCollection	Yes	No	
@odata.etag	As reported in GetResourceETag command.	No	No	This field contains common ETag, which is calculated skipping individual ETags.
@odata.context	N/A			Provided through GetSchemaURI
Status.health/ Status.healthRollupDescription	As reflected in health sensor of NIC (Card Composite State Sensor, <i>sensorID</i> = 5). Normal = "OK" Upper Non-Critical = "Warning" Upper Critical = "Critical" Upper Fatal = "Critical" If the Card Composite State Sensor (<i>sensorID</i> = 5) is uninitialized, Network Controller State sensor (<i>sensorID</i> = 60) is used instead. If this sensor is uninitialized, the operation fails.	No/Yes	No/No	Not applicable for collections.
Status.health/ Status.healthRollup	Same as "Status.health/ Status.healthRollupDescription" above.	No	No	Not applicable for collections.

10.8.6.5.3 ACD Schemas

10.8.6.5.3.1 Port v1.6.0

Note: Function-to-port mapping is linear. Port 0 represents the port that is mapped to function 0. The exposed ports are the ones for which PRTGEN_CNF.PORT_DIS is cleared.

Property	Value	Cacheable	Writable	Notes
Action				
.Reset				This action resets this port.
..target	%T<resource-ID>.0			
..title	#Port.Reset			
ActiveWidth	N/A	Yes	No	Integer. Currently not required.
CurrentSpeedGbps	Persistent location in NVM: Set the matching PHY types in TLV 0x134.	No	No	Integer value in Gb/s.
Ethernet		No	Yes	
.FlowControlConfiguration	Persistent location in NVM: Set the Link FC options in TLV 0x134.	No	Yes	Enum: None, TX, RX, TX_RX
.FlowControlStatus		No	No	Enum: None, TX, RX, TX_RX
.SupportedEthernetCapabilities	{None, EEE, WakeOnLAN}. EEE - either EEE LPI or EEE LLDP enabled, based on EMP <i>Capability</i> bit, PHY Capability, NVM EMP settings. WoL - PFPM_APM.APME	Yes	No	Array of enum: WakeOnLAN, EEE
.EEEEnabled	Is EEE enabled. Persistent location in NVM: <i>EEE Override Enable</i> field in Link Default Override Mask PFA TLV. Currently hard coded to FALSE.	No	No	Boolean.
.AssociatedMACAddress	An array of port associated MAC Addresses. Contains: - Factory LAN MAC Address - Current LAN MAC Address - Port MAC Address - PF LAA (MNG) MAC Address Only one copy of identical MAC Addresses should be present in array.	No	No	
.LLDPEnabled		No	Yes	Writable only in Setting Resource.
.LLDPReceive		No	No	
..ChassisId		No	No	
..ChassisIdSubtype		No	No	
..ManagementAddressIPv4		No	No	
..ManagementAddressIPv6		No	No	
..ManagementAddressMAC		No	No	
..ManagementVlanId		No	No	
..PortId		No	No	

Property	Value	Cacheable	Writable	Notes
..PortIdSubtype		No	No	
..LLDPTransmit		No	Yes	
..ChassisId		No	Yes	
..ChassisIdSubtype		No	Yes	
..ManagementAddressIPv4		No	Yes	
..ManagementAddressIPv6		No	Yes	
..ManagementAddressMAC		No	Yes	
..ManagementVlanId		No	Yes	
..PortId		No	Yes	
..PortIdSubtype		No	Yes	
FibreChannel	N/A			Fiber channel not supported.
GenZ	N/A			Gen-Z not supported.
Id	Resource offset.	Yes	No	
InterfaceEnabled	Port enabled or not.		Yes	Boolean
LinkConfiguration				Array of LinkConfiguration.
.AutoSpeedNegotiationCapable	Indicates whether the port has the capability to auto-negotiate speed.	Yes	No	Boolean
.AutoSpeedNegotiationEnable	Indicates whether the port is configured to auto-negotiate speed.	No	Yes	Boolean
.CapableLinkSpeedGbps	Set of link speed capabilities of this port.	Yes	No	Array of numbers.
.ConfiguredNetworkLinks	N/A			Array of ConfiguredNetworkLink. Currently not required.
..ConfiguredLinkSpeedGbps	N/A		Yes	Number. Currently not required
..ConfiguredWidth	N/A		Yes	Integer. Currently not required.
LinkNetworkTechnology	Ethernet		No	Enum: Ethernet, InfiniBand, FibreChannel, GenZ
LinkState		No	Yes	Enum: Enabled, Disabled
LinkStatus			Yes	Enum: LinkUp, Starting, Training, LinkDown, NoLink
LinkTransitionIndicator			Yes	Integer
Links	N/A			
Location	N/A			Currently not required.
LocationIndicatorActive	N/A			Currently not required.
MaxFrameSize	PRTMAC_HSEC_CTL_RX_MAX_PACKET_LEN Bits 14:0	Yes	No	
MaxSpeedGbps			No	Integer in Gb/s units.

Property	Value	Cacheable	Writable	Notes
Metrics	N/A			
Name	Device Name String + " Port Current Setting" for resources 100-107, Device Name String + " Port Pending Setting" for resources 110-117	Yes	No	
Oem				
PortId	The label of this port on the physical package for this port. Contains number only aligned to labeling starting from "1" for first port.	Yes	No	String Currently not required. Note:
PortMedium	"Optical" when optical module in use otherwise "Electrical". Null for unpopulated cage.			Enum: Electrical, Optical
PortProtocol	Ethernet	Yes	No	Enum: PCIe, AHCI, UHCI, SAS, SATA, USB, NVMe, FC, iSCSI, FCP, FICON, NVMeOverFabrics, SMB, NFSv3, NFSv4, HTTP, HTTPS, FTP, SFTP, iWARP, RoCE, RoCEv2, I ² C, TCP, UDP, TFTP, GenZ, MultiProtocol, InfiniBand, Ethernet, OEM Currently not required.
PortType	Hard-coded BidirectionalPort	Yes	No	Enum: UpstreamPort, DownstreamPort, InterswitchPort, ManagementPort, BidirectionalPort, UnconfiguredPort
SignalDetected	N/A	No	No	Boolean Currently not required.
Status	"Enabled" or "Disabled" based on PRTGEN_STATUS.PORT_VALID value	No	No	
Width	N/A			Integer. Currently not required.
FunctionMaxBandwidth		No	Yes	
.AllocationPercent	The bandwidth allocated per function (100%).	No	Yes	Integer
.NetworkDeviceFunction	Link to <i>NetworkDeviceFunction</i> belonging to port that the bandwidth is referring to.	No	No	Link
FunctionMinBandwidth		No	Yes	
.AllocationPercent	The bandwidth allocated per function (100%).	No	Yes	Integer
.NetworkDeviceFunction	Link to <i>NetworkDeviceFunction</i> belonging to port that the bandwidth is referring to.	No	No	Link
LinkConfiguration				
.ConfiguredNetworkLinks				Array of values below.
..ConfiguredLinkSpeedGbps	Persistent location in NVM: Set the matching PHY types in TLV 0x134.	No	Yes	Number (real)

Property	Value	Cacheable	Writable	Notes
Metrics	Link to associated PortMetrics resource.	Yes	No	
@Redfish.Settings				
..@odata.type	"#Settings.v1_3_3.Settings"	Yes	No	
.SettingObject				
..@odata.id	bejResourceLink to Settings Resource (current ID + 10).			
.SupportedApplyTimes	"OnReset"	Yes	No	

10.8.6.5.3.2 PortCollection

Property	Source/Value	Cacheable	Writable	Notes
Name	"Ports"	Yes	No	
Members@odata.count	Size of Members array.	Yes	No	
Members	Resource IDs of all the exposed NetworkPort resources.	Yes	No	100-107 110-117

10.8.6.5.3.3 NetworkInterface v1.2.0

Property	Source	Cacheable	Writable	Notes
Links.NetworkAdapter	NetworkAdapter Resource ID.		No	
Name	"E610 Network Interface"	Yes	No	
NetworkDeviceFunctions	NetworkDeviceFunctionCollection	Yes	No	
NetworkPorts	N/A	Yes	No	Deprecated.
Ports	PortCollection	Yes	No	
Status.State	"StandbyOffline" - If in D3/DR state for all PFs. "Starting" - If in D0 state and host driver is not loaded. "Updating" - If NVM firmware update is in progress. "Enabled" - Otherwise.	No	No	
ID	"%I5"	Yes	No	Deferred binding flag must be set.

10.8.6.5.3.4 NetworkAdapter v1.7.0

Property	Source	Cacheable	Writable	Notes
Actions				Activate the flow of reset to defaults. Copies the Factory defaults back to PFA and Extended TLV (MinSrev - DCR111 should be kept from original PFA).
..#NetworkAdapter.ResetSettingsToDefault		Yes	No	
..target	"%T1.0"	Yes	No	

Property	Source	Cacheable	Writable	Notes
..title	"#NetworkAdapter.ResetSettingsToDefault"	Yes	No	
Assembly	N/A	Yes	No	Not supported.
Controllers				
ControllerCapabilities				
DataCenterBridging				
Capable	TRUE	Yes	No	
NPAR				
NparCapable	FALSE	Yes	No	The E610 does not support NPAR.
NPIV	N/A			
LLDPEnabled		No	Yes	Writable only in Setting Resource.
NetworkDeviceFunctionCount	Number of enabled PCIe functions.	Yes	No	
NetworkPortCount	Same as NetworkDeviceFunctionCount.			
VirtualizationOffload				
SRIOV				
SRIOVVEPACapable	TRUE	Yes	No	
VirtualFunction				
DeviceMaxCount	0x100	Yes	No	Max VFs in controller.
MinAssignmentGroupSize	0x1	Yes	No	Min VFs per PF.
NetworkPortMaxCount	0x100	Yes	No	Max VFs per port.
FirmwarePackageVersion	Firmware version (Manual), Get Version AQC	Yes	No	String.
Links				
NetworkDeviceFunctions	Array of NetworkDeviceFunction references.	Yes	No	
NetworkDeviceFunctions@odata.count	Number of NetworkDeviceFunctions elements.			
NetworkPorts	N/A	Yes	No	Deprecated.
Ports	PortCollection	Yes	No	
PCIeDevices	Array of PCIeDevices references.	Yes	No	No implemented.
PCIeDevices@odata.count	Number of PCIeDevices elements			
Location	N/A			Not exposed.
PCIeInterface				
LanesInUse	Read from PCIe Configuration space at Link Status Register at Negotiated Link Width.	Yes	No	
MaxLanes	16 = Big Die, 8 = Small Die. Read from GL_UFUSE_SOC[7]. 0 = 16 1 = 8	Yes	No	

Property	Source	Cacheable	Writable	Notes
MaxPCIeType	Config space of function 0 - Link capabilities register (0xAC) bits 3:0 - Maximum Link Speed.	Yes	No	
PCIeType	{enum:"Gen1","Gen2","Gen3","Gen4"} Read from PCIe Configuration space at Link Status Register at Current Link Speed. The field value indicates the Gen version.	Yes	No	
Manufacturer	"Intel Corp."	Yes	No	
Model	"E610"	Yes	No	Read default device ID from PFPCI_DEVID.PF_DEV_ID; xxAM2 from GL_UFUSE_SOC[3:2]
Name	"E610 Network Adapter"	Yes	No	
NetworkDeviceFunctions	NetworkDeviceFunctionCollection	Yes	No	
NetworkPorts	NetworkPortCollection	Yes	No	
PartNumber	Part Number (PN) is 11 byte value maintained in VPD.	Yes	No	
SKU	GL_UFUSE_SOC[3:2]: 0 = CAM2 1 = CAM1 2 = XXVAM2	Yes	No	
SerialNumber	Read from GLPCI_SERH/L.	Yes	No	
Status.State	"StandbyOffline" - If in D3/DR state for all PFs. "Starting" - If in D0 state and host driver is not loaded. "Updating" - If NVM firmware update is in progress. "Enabled" - Otherwise.	No	No	Same as in NetworkInterface schema.
Idnetifiers	N/A			
ID	"%I1"	Yes	No	Deferred binding flag must be set.
Metrics	Link to associated NetworkAdapterMetrics resource.	Yes	No	
@Redfish.Settings				
._odata.type	"#Settings.v1_3_3.Settings"	Yes	No	
.SettingObject				
..@odata.id	bejResourceLink to Settings Resource (current ID + 10)			
.SupportedApplyTimes	"OnReset"	Yes	No	

10.8.6.5.3.5 NetworkDeviceFunction v1.6.0

Note: Resource IDs 200-207 are RO. All changes are done through resource IDs 210-217. After reset (PCIR or EMPR), the content of 210-217 is applied and copied to 200-207. The exposed functions are the ones for which PFPCI_FUNC.FUNC_DIS is cleared.

Property	Source/Value	Cacheable	Writable	Notes
AssignablePhysicalNetworkPorts	Resource IDs of the ports available in this SKU (100-107).	Yes	No	
AssignablePhysicalNetworkPorts@odata.count	Size of array of physical port references that this network device function may be assigned to.	Yes	No	
BootMode	1 = PXE - If PXE enabled on this port. 0 = Disabled - Otherwise. PXE is enabled if NVM Setup Options PCI Function[n], DBS field, has value 0x0-0x2 (not Local only) Persistent location in NVM: PFA	Yes	Yes	
DeviceEnabled	PFPCI_STATUS1.FUNC_VALID Persistent location in NVM: PFA	Yes	Yes	This value is maintained by ANVM.
Ethernet.MACAddress	MAC Address of the port as assigned by NVM/Alt RAM/LAA. A write is written in ALT RAM Current LAN MAC Address (low/ high) entries and stored in NVM in PF MAC Addresses TLV in PFA. Read is from the LAA MAC Address as set by Manage MAC Address write AQ command (update LAA address) or ALT RAM if not set or NVM if Alt RAM is not valid.	No	Yes	
Ethernet.MTUSize	At init, 9710 (9728-18). May be modified by RDE. Persistent location in NVM: PFA TLV RDE	Yes	Yes	
Ethernet.MTUSizeMaximum	The E610's Maximum Transmit Unit Size (MTU).	Yes	No	Integer
Ethernet.PermanentMACAddress	MAC Address of the port as stored in factory default section in NVM.	Yes	No	
Ethernet.VLAN	N/A			
Ethernet.VLANs	N/A			
MaxVirtualFunctions	As reported in Virtual Function capability (0x0013) in Discover Function Capabilities (PF_VT_PFALLOC.LASTVF - PF_VT_PFALLOC.FIRSTVF + 1 if PF_VT_PFALLOC.VALID is set, 0 otherwise) Refer to firmware API CDB_GetPf2VfMapping usage.	Yes	No	
Name	"NetworkDeviceFunction Current Settings" for ResourceID 200-207. "NetworkDeviceFunction Pending Settings" for ResourceID 210-217.	Yes	No	
NetDevFuncCapabilities	Source: Always ["Ethernet"]	Yes	No	Array of enum.
NetDevFuncType	1 "Ethernet"	Yes	No	

Property	Source/Value	Cacheable	Writable	Notes
Status.State	"StandbyOffline" - If in D3/DR state. "Starting" - If in D0 state and host driver is not loaded. "Updating" - If NVM firmware update is in progress. "Enabled" - PFPCI_STATUS1.FUNC_VALID is set "Disabled" - Otherwise.	No	No	
VirtualFunctionsEnabled	Config space of the function - PCIe SR-IOV Control Register (0x168) bit 0- VF Enable.	No	No	
iSCSIBoot	N/A	Yes	No	Not supported.
@redfish.Settings				Reported only in resources 200-207
SettingsObject	Points to the next setting = Resource ID +10	No	No	
SupportedApplyTimes	["OnReset"]	No	No	
ID	Resource Offset	Yes	No	
Links				
PCIeFunction	The link to the <i>PCIeFunction</i> associated with this network device function.	Yes	No	
PhysicalNetworkPortAssignment	The link to <i>NetworkPort</i> to which this device function is assigned.	Yes	No	
Metrics	Link to associated NetworkDeviceFunctionMetrics resource	Yes	No	

10.8.6.5.3.6 NetworkDeviceFunctionCollection

Property	Source	Cacheable	Writable	Notes
Name	NetworkDeviceFunctions	Yes	No	
Members@odata.count	Size of Members array.	Yes	No	
Members	Resource IDs of all the exposed NetworkDeviceFunction resources.	Yes	No	200-207

10.8.6.5.3.7 PCIeDevice v1.4.0

Note: This schema is tied to the global schemas using deferred binding.

Property	Source/Value	Cacheable	Writable	Notes
AssetTag	AssetTag field in NVM RDE PFA TLV. Persistent location in NVM: PFA	Yes	Yes	Should be initiated by a PUT command- null in initial image.
DeviceType	Config space of function 0 - Header type register (0xE): 0x0 = 0 (SingleFunction) 0x80 = 1 (MultiFunction) otherwise	Yes	No	
FirmwareVersion	As reported in Get Version admin command.	Yes	No	
Manufacturer	"Intel Corp."	Yes	No	

Property	Source/Value	Cacheable	Writable	Notes
Model	"E610"	Yes	No	
Name	"E610"	Yes	No	
PCIeInterface.LanesInUse	Config space of function 0 - Link Status Register (0xB2) bits 9:4 - <i>Negotiated Link Width</i>	No	No	
PCIeInterface.MaxLanes	Config space of function 0 - Link Capabilities Register (0xAC) bits 9:4 - <i>Maximum Link Width</i>	Yes	No	
PCIeInterface.MaxPCIeType	Config space of function 0 - Link Capabilities Register (0xAC) bits 3:0 - <i>Maximum Link Speed</i>	Yes	No	Subtract 1 from read value to get the enum value.
PCIeInterface.PCIeType	Config space of function 0 - Link Status Register (0xB2) bits 3:0 - <i>Current Link Speed</i>	No	No	
PartNumber	Part Number (PN) is 11 byte value maintained in VPD.	Yes	No	
SKU	GL_UFUSE_SOC[3:2]: 0 = CAM2 1 = CAM1 2 = XXVAM2	Yes	No	
Serial Number	Config space of function 0 - Serial Number Registers (0x154:0x158)	Yes	No	Serial Number is calculated from MAC Address (it is MAC completed with 0xFF on 5th and 6th byte). Example: MAC: 00-A0- C9-23-45-67 SN: 00-A0-C9- FF-FF-23-45-67
Status.State	"StandbyOffline" - If in D3/DR state in all functions. "Starting" - If in D0 state and host driver is not loaded in at least one function and others are in D3. "Updating" - If NVM firmware update is in progress. "Enabled" - Otherwise.	No	No	
PCIeFunctions	Link to PCIeFunctionCollection.	Yes	No	
ID				

10.8.6.5.3.8 PCIeFunctionCollection

Property	Source/Value	Cacheable	Writable	Notes
Name	PCIeFunctions	Yes	No	
Members@odata.count	Size of Members array.	Yes	No	
Members	Resource IDs of all the exposed PCIeFunction resources.	Yes	No	300-307

10.8.6.5.3.9 PCIeFunction v1.2.3

Note: Only Physical functions are exposed through RDE.

Property	Source/Value	Cacheable	Writable	Notes
ClassCode	"0x020000" ("EthernetController")	Yes	No	
DeviceClass	NetworkController	Yes	No	
DeviceID	PFPCI_DEVID.PF_DEV_ID (presented in hexadecimal format)	Yes	No	
FunctionType	0 (Physical)	Yes	No	
FunctionID	The PCIe Function Number of this PF.	Yes	No	Integer
Name	"E610"	Yes	No	
RevisionID	GLPCI_DREVID XOR GLPCI_REVID	Yes	No	
Status.State	"StandbyOffline" - If in D3/DR state. "Starting" - If in D0 state and host driver is not loaded. "Updating" - If NVM firmware update is in progress. "Enabled" - Otherwise	Yes	No	
SubsystemID	PFPCI_SUBSYSID.PF_SUBSYS_ID	Yes	No	
SubsystemVendorId	GLPCI_SUBVENID (presented in hexadecimal format)	Yes	No	
VendorID	GLPCI_VENDORID.VENDORID (presented in hexadecimal format)	Yes	No	
ID	Resource Offset	Yes	No	

10.8.6.5.4 Metrics Schemas

10.8.6.5.4.1 NetworkAdapterMetrics

Property	Value	Cacheable	Writable	Notes
Description	"Usage and health statistics for Ethernet adapter %I1"	Yes	No	String
Id	"E610 Network Adapter Metrics"	Yes	No	
NCSIRXBytes	N/A	No	No	
NCSIRXFrames	Number of NC-SI frames received since reset, including both pass-through and non-pass-through traffic.	No	No	
NCSITXBytes	N/A	No	No	
NCSITXFrames	Number of NC-SI frames sent since reset, including both pass-through and non-pass-through traffic.	No	No	
Name	"E610 Network Adapter Metrics"	Yes	No	
RXBytes	GLPRT_GORCL/H over all ports	No	No	
RXMulticastFrames	GLPRT_MPRCL/H over all ports	No	No	
RXUnicastFrames	GLPRT_UPRCL/H over all ports	No	No	
TXBytes	GLPRT_GOTCL/H over all ports	No	No	
TXMulticastFrames	GLPRT_MPTCL/H over all ports	No	No	
TXUnicastFrames	GLPRT_UPTCL/H over all ports	No	No	

10.8.6.5.4.2 PortMetrics

Property	Value	Cacheable	Writable	Notes
Description	"Usage and health statistics for Ethernet port x", where x is port ID.	Yes	No	String
Id	"Metrics"	Yes	No	
Name	"E610 Port Metrics"	Yes	No	
Newtorking		No	No	
+RDMARXBytes	From RDMA General Purpose Registers	No	No	
+RDMARXRequests	From RDMA General Purpose Registers	No	No	
+RDMATXBytes	From RDMA General Purpose Registers	No	No	
+RDMATXRequests	From RDMA General Purpose Registers	No	No	
+RXBroadcastFrame	GLPRT_BPRCL/H	No	No	
+RXDiscards	GLPRT_RDPC	No	No	
+RXFCSErrors	GLPRT_CRCERRS	No	No	
+RXFalseCarrierErrors	N/A	No	No	Not Supported
+RXFrameAlignmentErrors	N/A	No	No	Not Supported
+RXFrames	RXBroadcastFrame + RXMulticastFrames + RXUnicastFrames	No	No	
+RXMulticastFrames	GLPRT_MPRCL/H	No	No	
+RXOversizeFrames	GLPRT_ROC	No	No	
+RXPFCErrors	Sum of all the port's GLPRT_PXONRXCNT and GLPRT_PXOFFRXCNT counters.	No	No	
+RXPauseXOFFFrames	GLPRT_LXOFFRXCNT	No	No	
+RXPauseXONFrames	GLPRT_LXONRXCNT	No	No	
+RXUndersizeFrames	GLPRT_RUC	No	No	
+RXUnicastFrames	GLPRT_UPRCL/H	No	No	
+TXBroadcastFrames	GLPRT_BPTCL/H	No	No	
+TXDiscards	N/A	No	No	Not Supported
+TXFrames	TXBroadcastFrame + TXMulticastFrames + TXUnicastFrames	No	No	
+TXMulticastFrames	GLPRT_MPTCL/H	No	No	
+TXPFCErrors	Sum of all the port's GLPRT_PXONTXCNT and GLPRT_PXOFFTXCNT counters.	No	No	
+TXPauseXOFFFrames	GLPRT_LXOFFTXCNT	No	No	
+TXPauseXONFrames	LXONTXCNT	No	No	
+TXUnicastFrames	GLPRT_UPTCL/H	No	No	
RXBytes	GLPRT_GORCL/H	No	No	
RXErrors	Sum of: GLPRT_CRCERRS, GLPRT_ILLERRC, GLPRT_MLFC, GLPRT_MRFC, GLPRT_RLEC, GLPRT_RUC, GLPRT_RFC, GLPRT_ROC, GLPRT_RJC	No	No	
TXBytes	GLPRT_GOTCL/H	No	No	

Property	Value	Cacheable	Writable	Notes
TxErrors	N/A	No	No	
Transceivers	N/A	No	No	

10.8.6.5.4.3 NetworkDeviceFunctionMetrics

Property	Value	Cacheable	Writable	Notes
Description	Usage and health statistics for Device Function x", where x is the ID of the PF.	No	No	String
Ethernet		No	No	
+NumOffloadedIPv4Conns	N/A	No	No	
+NumOffloadedIPv6Conns	N/A	No	No	
Id	"Metrics"	Yes	No	
Name	"Network Device Function Metrics"	Yes	No	String
RXBytes	Same as PortMetrics.RXbytes	No	No	
RXFrames	Same as PortMetrics.Networking.RXFrames	No	No	
RXMulticastFrames	Same as PortMetrics.Networking.RXMulticastFrames	No	No	
RXQueuesEmpty	N/A	No	No	
RXQueuesFull	N/A	No	No	
RXUnicastFrames	Same as PortMetrics.Networking.RXUnicastFrames	No	No	
TXBytes	Same as PortMetrics.TXbytes	No	No	
TXFrames	Same as PortMetrics.Networking.TXFrames	No	No	
TXMulticastFrames	Same as PortMetrics.Networking.TXMulticastFrames	No	No	
TXQueuesEmpty	N/A	No	No	
TXQueuesFull	N/A	No	No	
TXUnicastFrames	Same as PortMetrics.Networking.TXUnicastFrames	No	No	

10.8.6.5.5 Non-ACD Schemas

10.8.6.5.5.1 EthernetInterface v1.5.1

Note: Resource IDs 400-407 are RO. All changes are done through resource IDs 410-417. After reset (PCIR or EMPR), the content of 410-417 is applied and copied to 400-407.

Property	Source	Cacheable	Writable	Notes
Actons	N/A			
AutoNeg	N/A	N/A	N/A	Not part of OCP profile - and no support in link management.
DHCPv4	N/A			
DHCPv6	N/A			
FQDN	N/A			
FullDuplex	"True"	Yes	No	Bool
HostName	N/A			
IPv4Addresses	N/A (external)			
IPv4StaticAddresses	N/A (external)			
IPv6AddressPolicyTable	N/A (external)			
IPv6Addresses	N/A (external)			
IPv6DefaultGateway	N/A (external)			
IPv6StaticAddresses	N/A (external)			
IPv6StaticDefaultGateways	N/A (external)			
InterfaceEnabled	PRTGEN_STATUS.PORT_VALID Persistent location in NVM: PFA	Yes	Yes	For enabled port.
LinkStatus	Link API: Link_is_up	No	No	{'LinkDown', 'LinkUp', 'NoLink'}
Links				
Chassis	N/A (external)			
Endpoints	N/A (analog to PF, not part of OCPBaselineHardwareManagent profile)			
Endpoints@odata.count	N/A			
HostInterface	N/A (external)			
Oem	N/A			
MACAddress	MAC Address of the port as assigned by NVM/ Alt RAM/LAA. A write is written in ALT RAM Current LAN MAC Address (low/high) entries and stored in NVM in PF MAC Addresses TLV in PFA. Read is from the LAA MAC Address as set by Manage MAC Address write AQ command (update LAA address) or ALT RAM if not set or NVM if Alt RAM is not valid.	Yes	Yes	
MTUSize	At init, 9710 (9728-18). May be modified by RDE. Persistent location in NVM: PFA TLV RDE	Yes	Yes	

Property	Source	Cacheable	Writable	Notes
MaxIPv6StaticAddresses	N/A ("null")			
Name	"E610 Ethernet Interface Current Settings" for ResourceID 400-407. "E610 Ethernet Interface Pending Settings" for ResourceID 410-417.	Yes	No	
NameServers	N/A, Stack level (DHCP related)			
Oem	N/A			
PermanentMACAddress	MAC Address of the port as reflected in NVM factory default section.	Yes	No	
SpeedMbps	Link API: Link_get_speed Persistent location in NVM: Per Port - Speeds[n] (0x004D + 2*n, n=0...7)	No	Yes	
StatelessAddressAutoConfig	N/A (external, IPv4 and IPv6)			
StaticNameServers	N/A, Stack level (DHCP related)			
Status.State	"Enabled"	No	No	
UefiDevicePath	N/A			
VLAN	N/A (external)			
VLANs	N/A ("null")	Yes	No	
@redfish.Settings.SettingsObject	Points to the next setting = Resource ID +10	No	No	Reported only if a pending configuration exists.
FallbackAddress	N/A			
@redfish.Settings				Reported only in resources 400-407
SettingsObject	Points to the next setting = Resource ID +10	No	No	
SupportedApplyTimes	["OnReset"]	No	No	
ID	Resource Offset	Yes	No	

10.8.6.5.5.2 EthernetInterfaceCollection

Property	Source	Cacheable	Writable	Notes
Name	"EthernetInterfaces"	Yes	No	
Members@odata.count	Size of Members array.	Yes	No	
Members	Resource IDs of all the exposed EthernetInterfaces resources.	Yes	No	400-407, 410-417

10.8.6.5.6 Generic Schemas

10.8.6.5.6.1 Event

Note: The Event schema is used to encode events in BEJ.

Property	Notes
Actions	N/A - Not required.
Context	N/A - Not required.
Description	N/A - Not required.
Events	Array of <i>EventRecords</i> .
.Actions	N/A - Not required.
.Context	N/A - Not required.
.EventGroupId	N/A - Not required. Can be used to group together events with same root cause.
.EventId	N/A - Not required.
.EventTimestamp	N/A - Not required.
.EventType	Deprecated. Enum value "Alert".
.MemberId	Event member element ID within the Events array.
.Message	N/A - Not required.
.MessageArgs	Array of strings containing the arguments needed for the Registry message.
.MessageId	The identifier for the message. Must be a string formatted as: RegistryName.MajorVersion.MinorVersion.MessageKey See DSP0266 <i>MessageID</i> format section. Example: NetworkDevice.1.0.CableInserted Note: If BEJ 1.1 is supported (by BMC and firmware) this is <i>bejRegistryItem</i> .
.MessageSeverity	N/A - Not required.
.Oem	N/A - Not required.
.OriginOfCondition	Reference to related triggering resource.
.Severity	N/A - Not required and deprecated.
.SpecificEventExistsInGroup	N/A - Not required.
Events@odata.count	Number of events in Event array
ID	"1"
Name	"Event"
OEM	N/A - Not required.

10.8.6.5.6.2 NetworkDevice Registry v1.0.1

Note: Registry messages that are not listed below are not required.

Message	Parameters	Trigger
CableInserted	1 = The `Id` of the network adapter. 2 = The `Id` of the network port.	Cable inserted into port. Currently not supported
CableRemoved	1 = The `Id` of the network adapter. 2 = The `Id` of the network port.	Cable removed from port. Currently not supported
ConnectionDropped	1 = The `Id` of the network adapter. 2 = The `Id` of the network port. 3 = The `Id` of the network function.	Link down.
ConnectionEstablished	1 = The `Id` of the network adapter. 2 = The `Id` of the network port. 3 = The `Id` of the network function.	Link up.
DegradedConnectionEstablished	1 = The `Id` of the network adapter. 2 = The `Id` of the network port. 3 = The `Id` of the network function.	Link established on a lower speed than is supported by both the device and the Link Partner. Currently not supported
LinkFlapDetected	1 = The `Id` of the network adapter. 2 = The `Id` of the network port. 3 = The `Id` of the network function. 4 = The number of times the link has flapped. 5 = The number of minutes over which the link has flapped.	Network connection has repeatedly been established and dropped.

10.8.6.5.7 Custom OEM Schema Extensions

Custom OEM extensions to DMTF schemas as well as entirely custom OEM schemas are not described here.

The presence of OEM RDE extensions in an NVM image, if any, are indicated by the non-default value of the IANA field. IANA is maintained in PFA OEM section (for more information, see [Section 6.3.50.10](#)).

10.8.6.5.8 Profiles

Each profile defines a specific set of supported schemas. The ACD profile is supported and all the non-ACD schemas have enable bits inside the NVM.

The AdvancedCommunicationDevice profile is described here: https://www.dmtf.org/sites/default/files/Redfish_School-Advanced_Communication_Devices_%28ACD%29.pdf

10.8.6.6 Dictionaries

Dictionaries are stored in signed NVM (not in shadow RAM).

Each of the Major Schema has associated with it an individual Major dictionary. Three additional dictionaries contain definitions of common properties.

Dictionaries must contain definitions for decoding all BEJ tuples that can be generated by this implementation of RDE. All properties that might be encountered during an UPDATE or REPLACE request, but which cannot be ignored, must also have corresponding dictionary definitions. Properties that are ignored on both reads and writes serve no purpose and should be excluded from dictionaries.

10.8.6.6.1 Dictionary Storage in NVM

The Dictionaries are stored at a 4K boundary within the NVM signed area and pointed by the RDE Dictionaries pointer at offset 0x5F in the init module.

The allocated size in NVM map 32K bytes. The format is as follows:

Table 10-76. RDE Dictionaries Format in NVM

Word Offset ¹	Description	Remarks
0x0000	Length	Total section length.
0x0001	Number of dictionaries	13 - For read-only support.
0x0002 + n*5	Resource ID Low	
0x0003 + n*5	Resource ID High	
0x0004 + n*5	Schema Class (Bits[7:0])	Bits[15:8] are reserved
0x0005 + n*5	Schema dictionary offset	If MSB=1, the resolution of the value is 1 page. If MSB=0, the resolution of the value is one WORD.
0x0006 + n*5	Schema URI offset	

1. n = Dictionary index in range from 0 to Number of dictionaries.

Schema dictionary offset points to following structure:

Word Offset	Description	Remarks
0x0000	Section length	Words.
0x0001...	Dictionary	Binary blob.

Dictionaries have a binary format according to DSP0218 standard (Section 7.2.3.2, "Dictionary binary format").

Schema URI offset points to following structure:

Word Offset	Description	Remarks
0x0000	Section length	In bytes.
0x0001...	URI	String.

Schema URI is stored as a string. The varstring assumes an ASCII encoding. The length is derived from the section length.

10.9 OCP NIC 3.0 Support

The OCP 3.0 specification defines a list of manageability features that are needed to be compliant with the specification. The following features are included in the OCP NIC 3.0 support in the E610:

Table 10-77. OCP NIC 3.0 Support

Feature	Notes
NC-SI 1.1	See Section 10.6 .
MCTP 1.3	See Section 10.8 .
MAC Address Provisioning	Requires NC-SI 1.2. Assumes current draft. May change.
Temperature Reporting - ASIC	See Section 10.7.4.1 .
Temperature Reporting - Modules	See Section 10.7.4.1 .
SFP/QSFP Modules Power Reporting	See Section 10.7.4.1 .
Board Power Reporting	FRU over I ² C. Defined at the board level.
Firmware Inventory and Update	See Section 10.8.4 .
Secure Firmware	See Section 3.3.9 .
NC-SI Package Addressing	See Section 10.3.2.2.1 .
FAN_ON_AUX Pin	See Section 10.9.1 .

10.9.1 Support for FAN_ON_AUX Pin

Firmware flow at init is:

1. Check that the FAN_ON_AUX valid bit is set. If it is not, skip its initialization.
2. Check that the FAN_ON_AUX SDP is according SDP hardware range. If it is not, report error in GL_MNG_FWSM.EXT_ERR_IND and skip its initialization (report error, but do not enter error flow).
3. Check that the FAN_ON_AUX SDP is configured as Output,. If it is not, report error in GL_MNG_FWSM.EXT_ERR_IND and skip its initialization (report error, but do not enter error flow).
4. Configure the FAN_ON_AUX SDP with the FAN_ON_AUX bit value from NVM.

10.10 SPDM Support

The *Security Protocol and Data Model (SPDM) Specification* defines messages, data objects and sequences for performing security message exchanges between two devices within a platform over MCTP. The message exchanges defined in SPDM includes authentication of hardware identities and measurement for firmware identities as well as device configuration and state.

SPDM and its accompanying documents are DMTF specifications:

- DSP0274 defines the basic SPDM messages.
- DSP0275 defines the SPDM over MCTP binding and defines SPDM as MCTP type 5.
- DSP0276 defines how secure messages are sent as part of a SPDM secure session and maps them to MCTP type 6.
- DSP0277 defines how MCTP messages are secured by using SPDM sessions.

The E610 supports version 1.1.1b of DSP0274 (SPDMVersion V1.11 = 0x11) and version 1.0.0 of DSP0275, DSP0276, and DSP0277.

10.10.1 SPDM Enablement

SPDM settings are located in the SPDM Control Word in the Manageability Module in the NVM. SPDM functionality requires that the *SPDM_Enable* bit in the NVM is set to 1b. If set to 0b, the device will not support the MCTP message types corresponding to SPDM.

In addition, the Unique Device Secret must be provisioned in fuses. If the UDS was not provisioned into the device during manufacturing then SPDM functionality shall be disabled, and the device will not support the MCTP message types corresponding to SPDM.

If SPDM is enabled, it must be supported by both the main firmware image and the recovery firmware. It is not supported by the mini-loader or the boot ROM. If SPDM is being run from the recovery image, Measurement 4 will be a measure of the recovery firmware image only and not a measurement of the entire NVM bank.

10.10.2 Secure Session Limit

The device supports a maximum of one SPDM secure session on each supported interface. The two supported interfaces are the PCIe DOE (Data Object Exchange) mailbox and the SPDM over MCTP interface. Any attempts to create a new session before terminating the previous one on the same interface shall fail with a SessionLimitExceeded error.

10.10.3 Measurement Algorithm

The measurement algorithm for signed firmware images is always the hash algorithm used during the signature verification process. the E610 does not support different hash algorithms for signing and SPDM for a given image. Because the "MeasurementHashAlgo" from the NEGOTIATE_ALGORITHMS exchange may not necessarily be used, the hardware register field indicating the DMTF Spec Measurement Value Type indicates a Bitstream rather than a Digest. Nevertheless, firmware should always advertise SHA2-384 as the MeasurementHashAlgo. See [Section 10.10.5.7](#) for details.

10.10.4 Supported SPDM Commands

Table 10-78 lists the SPDM messages and their support in the E610 if SPDM is enabled:

Table 10-78. SPDM Messages

Request	Code Value	Response	Code Value	Support in the E610	Message Format
GET_DIGESTS	0x81	DIGESTS	0x01	Yes	Section 10.10.5.1
GET_CERTIFICATE	0x82	CERTIFICATE	0x02	Yes	
CHALLENGE	0x83	CHALLENGE_AUTH	0x03	Yes	Section 10.10.5.2
GET_VERSION	0x84	VERSION	0x04	Yes	Section 10.10.5.4
GET_MEASUREMENTS	0xE0	MEASUREMENTS	0x60	Yes	Section 10.10.5.5
GET_CAPABILITIES	0xE1	CAPABILITIES	0x61	Yes	Section 10.10.5.6
NEGOTIATE_ALGORITHMS	0xE3	ALGORITHMS	0x63	Yes	Section 10.10.5.7
KEY_EXCHANGE	0xE4	KEY_EXCHANGE_RSP	0x64	Yes	Section 10.10.5.8
FINISH	0xE5	FINISH_RSP	0x65	Yes	Section 10.10.5.9
PSK_EXCHANGE	0xE6	PSK_EXCHANGE_RSP	0x66	No	
PSK_FINISH	0xE7	PSK_FINISH_RSP	0x67	No	
HEARTBEAT	0xE8	HEARTBEAT_ACK	0x68	No	
KEY_UPDATE	0xE9	KEY_UPDATE_ACK	0x69	No	
GET_ENCAPSULATED_REQUEST	0xEA	ENCAPSULATED_REQUEST	0x6A	No	
DELIVER_ENCAPSULATED_RESPONSE	0xEB	ENCAPSULATED_RESPONSE_ACK	0x6B	No	
END_SESSION	0xEC	END_SESSION_ACK	0x6C	Yes	Section 10.10.5.10
VENDOR_DEFINED_REQUEST	0xFE			No	
RESPOND_IF_READY	0xFF			Yes	Section 10.10.5.11
		ERROR	0x7F	Yes	

10.10.5 SPDM Command Details

10.10.5.1 Get Digests (0x81)

Type	Request Data
uint8	SPDMVersion = 0x11
uint8	RequestResponseCode (GET_DIGESTS = 0x81)
uint8	Param1 - Reserved
uint8	Param2 - Reserved
Type	Response Data
uint8	SPDMVersion = 0x11
uint8	RequestResponseCode (DIGESTS = 0x1)
uint8	Param1 - Reserved
uint8	Param2 - Reserved
uint8*H	Digest[0]
...	
uint8*H	Digest[n-1]

10.10.5.2 Get Certificate (0x82)

Type	Request Data
uint8	SPDMVersion = 0x11
uint8	RequestResponseCode (GET_CERTIFICATE = 0x82)
uint8	Param1 - Slot number of the target certificate chain to read from. The value in this field shall be between 0 and 7 inclusive.
uint8	Param2 - Reserved
uint16	Offset - Offset in bytes from the start of the certificate chain to where the read request message begins. The Responder should send its certificate chain starting from this offset. For the first GET_CERTIFICATE request for a given slot, the Requester shall set this field to 0. For subsequent requests, Offset is set to the next portion of the certificate in that slot.
uint16	Length - Length of certificate chain data, in bytes, to be returned in the corresponding response. Length is an unsigned 16-bit integer. This value is the smaller of the following values: a. Capacity of the internal buffer of the Requester for receiving the certificate chain of the Responder. b. The RemainderLength of the preceding GET_CERTIFICATE response. If offset=0 and length=0xFFFF, the Requester is requesting the entire chain.
Type	Response Data
uint8	SPDMVersion = 0x11
uint8	RequestResponseCode (CERTIFICATE = 0x2)
uint8	Param1 - Slot number of the certificate chain returned.
uint8	Param2 - Reserved
uint16	PortionLength - Number of bytes of this portion of certificate chain. This should be less than or equal to Length received as part of the request. For example, the Responder might set this field to a value less than Length received as part of the request due to limitations on the internal buffer of the Responder
uint16	RemainderLength - Number of bytes of the certificate chain that have not been sent yet after the current response. For the last response, this field shall be 0 as an indication to the Requester that the entire certificate chain has been sent.
uint8* PortionLength	CertChain - Requested contents of target certificate chain

10.10.5.3 Challenge (0x83)

Type	Request Data
uint8	SPDMVersion = 0x11
uint8	RequestResponseCode (CHALLENGE = 0x83)
uint8	Param1 - Slot Number = 0 Return an Error message if Slot Number not equal 0
uint8	Param2 - Requested Measurement Summary Hash Type Return an Error message if not equal 0, 1 or 0xFF
uint32	Requester Nonce
Type	Response Data
uint8	SPDMVersion = 0x11
uint8	RequestResponseCode (VERSION = 0x3)
uint8	Param1 - Slot Number = 0

uint8	Param2 - Mask = 1
uint384	CertChainHash = Hash of the device certificate chain, or of the public key. If the device has advertised CERT_CAP, then this field shall contain the hash of the certificate chain that would be returned by the device in response to a GET_CERTIFICATE command. If the device has not advertised CERT_CAP, then this field shall contain a hash of the Device ID public key. This value is inserted by the firmware.
uint32	Nonce = A random number generated using the Secure Boot Block's TRNG and inserted by the SBB as part of generating the response.
uint384	This field is present only if <i>MeasurementSummaryHash</i> 1 or 0xFF If <i>MeasurementSummaryHash</i> is 1, the combined measurement of the TCB is requested. This includes the Hardware Measurement, the SBB Fuse Measurement, and the Mini-Loader Measurement, as these are in the TCB for the generation of higher-level measurements. TCB_measurement = hash(HW_measurement_bitmap, hash(SBB_fuses), hash(MiniLoader)). If Measurement SummaryHash is 0xFF, then the response requires the overall measurement. ALL_measurement = hash(HW_measurement_bitmap, hash(SBB_fuses), hash(MiniLoader), hash(NVM_Bank)). In both cases, the measurements are inserted by the SBB as part of generating the response.
uint16	OpaqueLength = 0 (no OpaqueData for now)
uint768	TPM_ALG_ECDSA_ECC_NIST_P384 based signature as defined in Section 10.9.1 of DSP0274 1.0.1 spec. Note that this signature is dynamic and depends on the history of commands received as described in Section 10.9.2.1 of DSP0274 1.0.1 spec. The possible sequences the E610 supports are: • GET_VERSION, GET_CAPABILITY, NEGOTIATE_ALGORITHMS, CHALLENGE • CHALLENGE

10.10.5.4 Get Version (0x84)

Type	Request Data
uint8	SPDMVersion = 0x11
uint8	RequestResponseCode (GET_VERSION = 0x84)
uint8	Param1 - Reserved
uint8	Param2 - Reserved
Type	Response Data
uint8	SPDMVersion = 0x11
uint8	RequestResponseCode (VERSION = 0x4)
uint8	Param1 - Reserved
uint8	Param2 - Reserved
uint8	VersionNumberEntryCount = 1
uint16	VersionNumberEntry1 = 0x1110 (1.1.1.0)

10.10.5.5 Get Measurements (0xE0)

Type	Request Data
uint8	SPDMVersion V1.1 = 0x11
uint8	RequestResponseCode (GET_MEASUREMENTS - 0xE0)
uint8	Param1 - Request attributes. See the GET_MEASUREMENTS request attributes table. (TBD)

uint8	Param2 - Measurement operation. A value of 0x0 queries the Responder for the total number of measurement blocks available. A value of 0xFF requests all measurement blocks. The number of supported measurements is 5 and out-of-range requests shall fail. A value between 0x1 and 0x05, inclusively, requests the measurement block at the index corresponding to that value.
uint8*32	Nonce The Requester should choose a random value. This field is only present if a signature is required on the response. See the GET_MEASUREMENTS request attributes table.
uint8	SlotIDParam Bit[7:4] = Reserved. Bit[3:0] = SlotID - Slot number of the certificate chain of the Responder that shall be used for authenticating the measurement(s). If the device did not advertise CERT_CAP, then the requester shall send 0xF in this field. This field is only present if a signature is required on the response. See the GET_MEASUREMENTS request attributes table.
Type	Response Data
uint8	SPDMVersion V1.1 = 0x11
uint8	RequestResponseCode (MEASUREMENTS = 0x60)
uint8	Param1 When <i>Param2</i> in the requested measurement operation is 0, this parameter returns the total number of measurement indices on the device, which is four. Otherwise, this field is reserved.
uint8	Param2 Bit[7:4] = Reserved. Bit[3:0] = SlotID - If this message contains a signature, this field contains the slot number of the certificate chain specified in the GET_MEASUREMENTS request, or 0xF if the Responder's public key was provisioned to the Requester previously. If this message does not contain a signature, this field is set to 0x0.
uint8	NumberOfBlocks Number of measurement blocks (N) in MeasurementRecord. If <i>Param2</i> in the requested measurement operation is 0, this field is 0.
uint8	MeasurementRecordLength Size of the <i>MeasurementRecord</i> field in bytes. If <i>Param2</i> in the requested measurement operation is 0, this field is 0.
uint8*L	MeasurementRecord Concatenation of all measurement blocks that correspond to the requested Measurement operation. Table 10-79 defines the measurement block structure for this device, as well as the possible measurements. L = <i>MeasurementRecordLength</i> Note:
uint8*32	Nonce The Responder should choose a random value. This value shall be populated by the SBB, using the SBB's TRNG.
uint8*2	OpaqueLength Size of the <i>OpaqueData</i> field in bytes. The value must not be greater than 1024 bytes. The device shall return zero in this field.
uint8*O	OpaqueData Free-form field, if present. The Responder may include Responder-specific information and/or information defined by its transport. O = <i>OpaqueLength</i> This field is not returned by this device.
uint8*S	Signature of the GET_MEASUREMENTS request and MEASUREMENTS response messages, excluding the <i>Signature</i> field and signed using the device private key. The Responder uses the asymmetric signing algorithm it selected during the last ALGORITHMS response message to the Requester, and S is the size of that asymmetric signing algorithm output. This field is conditional.

Table 10-79 defines the measurement indexes supported by the device.

Table 10-79. Supported Measurement Indexes

Measurement Index	Description
1	Hardware Measurement This measurement is a bitmap collected and reported by the SBB. It indicates the security state of the device, including debug unlock state and debug strap values. See below for the specific format of this measurement block. This measurement is considered to be in the TCB for measurement generation. The contents of this measurement are routed via internal lines from the various device IPs to the SBB's measurement bus. The values are latched when routed to the SBB. When requested as a single signed measurement as part of a GET_MEASUREMENT command, this measurement cannot be modified even by compromised firmware.
2	SBB Fuse Measurement This measurement is a hash of all root-of-trust fuses, excluding any in-field programmable fuses, such as key revocation and security revision. It includes all KAK hashes in the Signature and OEM roots-of-trust and represents the expected state of the device with respect to its root key programming. This value is provided by the SBB. This measurement is considered to be in the TCB for measurement generation. When requested as a single signed measurement as part of a GET_MEASUREMENT command, this measurement cannot be modified even by compromised firmware.
3	Mini-Loader Measurement This measurement is a hash of the Mini-Loader binary as measured by the Boot ROM as part of secure boot. It is provided to the SBB by the immutable Boot ROM. This measurement is considered to be in the TCB for measurement generation. It is reported as a raw bit stream, as described in Section 10.10.3. When requested as a single signed measurement as part of a GET_MEASUREMENT command, this measurement cannot be modified even by compromised firmware.
4	NVM Bank (firmware) Measurement This measurement is a hash of the NVM Bank booted by the Mini-Loader as part of secure boot. It is provided to the SBB by the Mini-Loader. This measurement is not considered to be in the TCB for measurement generation. It is reported as a raw bit stream, as described in Section 10.10.3. When requested as a single signed measurement as part of a GET_MEASUREMENT command, this measurement cannot be modified even by compromised Main firmware.
5	Firmware Managed Security Settings and Fuses This measurement is a bitmap of various device fuse bits and configuration states managed by the device firmware. This measurement is provided by firmware as all these settings are changeable and managed by the firmware itself in some form. This measurement is not considered to be in the TCB for measurement generation.

Table 10-80 through Table 10-83 details each measurement block's definition, as returned by the device. Note that DMTF SPDM Measurement Blocks are unaligned structures and should be treated as such in firmware/software.

Table 10-80. Measurement 1

Offset (Byte)	Field	Size (Bytes)	Value
0	Index	1	1
1	Measurement Specification	1	1 = DMTF Measurement Specification
2	Measurement Size	2	7
4	DMTF Spec Measurement Value Type	1	Bit [7]: 1 (Raw Bit Stream) Bits [6:0]: 2 (Hardware Configuration)
5	DMTF Spec Measurement Value Size	2	4

Table 10-80. Measurement 1 [continued]

Offset (Byte)	Field	Size (Bytes)	Value
7	Hardware State	2	Bit [0]: Intel Debug Unlock (policy 2 or 4) has occurred in the current power cycle. The expected secure state is 0. Bit [1]: OEM Debug Unlock (policy 5) has occurred in the current power cycle. The expected secure state is 0. Bits [3:2]: Reserved. Always returned as 0. Bit [4]: VAD unlock state. Whether the Vendor Authorized Debug authentication has taken place in the current power cycle. The expected secure state is 0. Bit [5]: CSR Protection Enabled. Whether the HIF protects R/O registers from host access. The expected secure state value is 1. Bit [6]: Authentication Enabled. Whether the firmware's secure boot mechanism is enabled. The expected secure state value is 1. Bit [7]: A0 Debug Strap. The expected secure state value is 0. Bit [8]: VAD Enable. Whether Vendor Authorized Debug is enabled. The expected value depends on the part's fuse configuration, and may vary. Bit [9]: VAD Override. Whether the VAD mechanism is overridden by fuse. The expected secure state value is 0. If this value is 1, then the next bit must be 1 for the device to be in a secure state. Bit [10]: VAD Resecure. Whether the VAD mechanism override is by itself overridden by the re-secure fuse. The expected value depends on the VAD Override bit. If VAD Override is 1, then the expected value is 1. If VAD Override is 0, then this bit is a "don't care". Bit [11]: BFP Disable. Whether Blank Flash Programming mode is disabled by fuse. The expected value depends on the part's desired fuse configuration. For standard Intel devices, the expected value is 0, since BFP is enabled. Bit [12]: SREV In Fuses. Whether the security revision is stored in flash or fuses. The expected value depends on the part's fuse configuration, and may vary. Bit [13]: Boot Halt. Whether the firmware boot was halted via strap during the last firmware boot cycle. While performing a boot halt does not compromise or bypass the security of the firmware boot process it would be highly unexpected to see this in the field. As such, the expected secure state value is 0. Bits [15:14]: Reserved, Always returned as 0.

Table 10-81. Measurement 2

Offset (Byte)	Field	Size (Bytes)	Value
0	Index	1	2
1	Measurement Specification	1	1 = DMTF Measurement Specification
2	Measurement Size	2	51
4	DMTF Spec Measurement Value Type	1	Bit [7]: 0 (Digest) Bits [6:0]: 2 (Hardware Configuration)
5	DMTF Spec Measurement Value Size	2	48
7	SBB Fuse Hash	48	SHA2-384 value of the SBB fuses, including all KAK fuses, but excluding all revocation fuses and SREV fuses.

Table 10-82. Measurement 3

Offset (Byte)	Field	Size (Bytes)	Value
0	Index	1	3
1	Measurement Specification	1	1 = DMTF Measurement Specification
2	Measurement Size	2	51
4	DMTF Spec Measurement Value Type	1	Bit [7]: 1 (Raw bit stream) Bits [6:0]: 1 (Mutable firmware)
5	DMTF Spec Measurement Value Size	2	48
7	Mini-Loader Hash	48	SHA2-384 value of the Mini-Loader used during firmware boot.

Table 10-83. Measurement 4

Offset (Byte)	Field	Size (Bytes)	Value
0	Index	1	4
1	Measurement Specification	1	1 = DMTF Measurement Specification
2	Measurement Size	2	51
4	DMTF Spec Measurement Value Type	1	Bit [7]: 1 (Raw bit stream) Bits [6:0]: 1 (Mutable firmware)
5	DMTF Spec Measurement Value Size	2	48
7	NVM Bank Hash	48	SHA2-384 value of the signed NVM bank loaded during the last firmware boot. If the recovery firmware is invoked, this will be a measurement of the recovery firmware.

Table 10-84. Measurement 5

Offset (Byte)	Field	Size (Bytes)	Value
0	Index	1	5
1	Measurement Specification	1	1 = DMTF Measurement Specification
2	Measurement Size	2	67
4	DMTF Spec Measurement Value Type	1	Bit [7]: 1 (Raw bit stream) Bits [6:0]: 2 (Hardware configuration)
5	DMTF Spec Measurement Value Size	2	64
7	NVM Bank Hash	93	Signature Root of Trust Provisioning: Byte 0: [0]: EMP Instruction Memory R/O [1]: Fuse Programming Locked [2:7]: Reserved Bytes 1 through 32: Security Revision bits. Byte 33: [2:0]: SigRoT KAK Revocation Bits [7:3]: Reserved Bytes 34 through 64: SigRoT ISK revocation bits

10.10.5.6 Get Capabilities (0xE1)

Type	Request Data
uint8	SPDMVersion = 0x11
uint8	RequestResponseCode (GET_CAPABILITIES = 0xE1)
uint8	Param1 - Reserved
uint8	Param2 - Reserved
Type	Response Data
uint8	SPDMVersion = 0x11
uint8	RequestResponseCode (CAPABILITIES = 0x61)
uint8	Param1 - Reserved
uint8	Param2 - Reserved
uint8	Reserved
uint8	CTExponent = Provided by the firmware from the <i>CTExponent</i> field of the SPDM Control Word in the NVM Manageability Module. Note: This value defines the maximum time that the device may take to perform a SPDM command which requires cryptographic processing. For example: CHALLENGE or GET_MEASUREMENT. The value is calculated as 2 to the power of <i>CTExponent</i> and the units are microseconds. Thus, for example, a <i>CTExponent</i> value of 15 means a maximum time of 32,768 μ s ($\sim$ 33 μ s)
uint16	Reserved
uint32	Flags (any bits not explicitly specified are returned as zero by the device) Bit [0]: CACHE_CAP = 0 Bit [1]: CERT_CAP = See note below. Bit [2]: CHAL_CAP = 1 Bits [4:3]: MEAS_CAP = 10b Bit [6]: ENCRYPT_CAP = 1 Bit [7]: MAC_CAP = 1 Bit [9]: KEY_EX_CAP = 1 Bit [16]: PUB_KEY_ID_CAP = See note below. Note: If the device does not report CERT_CAP, it shall report PUB_KEY_ID_CAP. If it does report CERT_CAP, then PUB_KEY_ID_CAP shall be zero.
uint8	Reserved
uint8	Reserved
uint8	Reserved

10.10.5.7 Negotiate Algorithms (0xE3)

Type	Request Data
uint8	SPDMVersion = 0x11
uint8	RequestResponseCode (NEGOTIATE_ALGORITHMS = 0xE3)
uint8	Param1 - Reserved
uint8	Param2 - Reserved
uint8	Length of the entire request message, in bytes. Length must be less than or equal to 128 bytes.
uint8	MeasurementSpecification = 1 (DMTF Type) Return Error if value different than 1

uint8	Reserved
uint32	BaseAsymAlgo If byte 0 bit 7 (TPM_ALG_ECDSA_ECC_NIST_P384) is not set, then no common algorithm exists.
uint32	BaseHashAlgo If byte 0 bit 4 (TPM_ALG_SHA_384) is not set, then no common algorithm exists.
uint32*3	Reserved
uint8	ExtAsymCount = A Note: Requesters may specify extended algorithms, but this device does not support any.
uint8	ExtHashCount = E Note: Requesters may specify extended algorithms, but this device does not support any.
uint16	Reserved
uint32*A	ExtAsym (Ignored by this device.)
uint32*E	ExtHash (Ignored by this device.)
Variable	The <i>ReqAlgStruct</i> provided by the requester. See the SPDm specification (DSP0274) of the supported version for details. The requester will specify all algorithms they support, which can be an extensive list. This device only looks for two specific algorithms: AlgType = 0x2 (DHE = Diffie-Hellman Exchange), Alg = secp384r1 (ECDHE P384). AlgType = 0x3 (AEAD = Authenticated Encryption with Association Data), Alg = AES-256-GCM. No other algorithms, of any type, will be chosen.
Type	Response Data
uint8	SPDMVersion = 0x11
uint8	RequestResponseCode (ALGORITHMS = 0x63)
uint8	Param1 - Reserved
uint8	Param2 - Reserved
uint8	Length of the entire response message, in bytes. Length must be less than or equal to 128 bytes. In this device, the response length is 44 bytes.
uint8	MeasurementSpecificationSel = 1 (DMTF Type)
uint8	Reserved
uint32	MeasurementHashAlgo = 0x2 (TPM_ALG_SHA_384)
uint32	BaseAsymSel If <i>BaseAsymAlgo</i> byte 0 bit 7 was set in the request, return 0x80. Otherwise, return 0x0.
uint32	BaseHashSel If <i>BaseHashAlgo</i> byte 0 bit 4 was set in the request, return 0x10. Otherwise, return 0x0.
uint32	Reserved
uint32	Reserved
uint32	Reserved
uint8	ExtAsymSelCount = 0
uint8	ExtHashSelCount = 0
uint16	Reserved
uint8	AlgType = 0x2 (DHE = Diffie Hellman Exchange)
uint8	Bits [7:4]:BytesNum = 0x2 Bits [3:0]:DHE_AlgCount = 0x1 if the requester specified AlgType = 0x2 and the requester's AlgSupported contains mask 0x10 (ECDHE secp184r1), 0x0 otherwise.
uint16	If DHE_AlgCount is 0x0, then this value is zero. If DHE_AlgCount is 0x1, then this value shall be set to 0x10.

uint8	AlgType = 0x3 (AEADCipherSuite)
uint8	Bits [7:4]:BytesNum = 0x2 Bits [3:0]:AEAD_Algorithm = 0x1 if the requester specified AlgType = 0x3 and the requester's AlgSupported contains mask 0x2 (AES-256-GCM), 0x0 otherwise.
uint16	If AEAD_Algorithm is 0x0, then this value is zero. If AEAD_Algorithm is 0x1, then this value shall be set to 0x2.

10.10.5.8 Key Exchange (0xE4)

Type	Request Data
uint8	SPDMVersion = 0x11
uint8	RequestResponseCode (KEY_EXCHANGE = 0xE4)
uint8	Param1 Requested MeasurementSummaryHash type: 0x0 = No measurement. 0x1 = TCB measurement hash. 0xFF = All measurements hash. All other values will result in an error.
uint8	Param2 The slot number of the certificate chain which is to be used for the authentication. If the device did not advertise CERT_CAP, then the requester needs to send 0xFF as the value. If the specified certificate slot is not populated, the device shall return an InvalidRequest error.
uint16	ReqSessionID - Two-byte requester session ID.
uint16	Reserved
uint8 * 32	RandomData - Requester-provided random data.
uint8 * 144	RequesterExchangeData Represents the ECDHE X and Y values in network byte order. Specifically, X is [0: C - 1] and Y is [C : D - 1]. X is contained in bytes 0 through 95, Y is contained in bytes 96 through 143.
Type	Response Data
uint8	SPDMVersion = 0x11
uint8	RequestResponseCode (KEY_EXCHANGE_RSP = 0x64)
uint8	Param1 - HeartbeatPeriod Since this device does not support HEARTBEAT_CAP, this will be returned by the device as a zero.
uint8	Param2 - Reserved
uint16	RspSessionID The responder's chosen session ID. The overall Session ID is formed as a concatenation of the ReqSessionID and the RspSessionID.
uint8	MutAuthRequested Always returned as zero, as this device does not support mutual authentication.
uint8	SlotIDParam Always returned as zero, as this device does not support mutual authentication.
uint8 * 32	RspRandomData Responder provided random data, generated and provided by the SBB.
uint8 * 144	ResponderExchangeData The ECDHE public information generated by the responder. Represents the ECDHE X and Y values in network byte order. Specifically, X is [0: C - 1] and Y is [C : D - 1]. X is contained in bytes 0 through 95, Y is contained in bytes 96 through 143.

uint384	This field is present only if MeasurementSummaryHash in the request was 1 or 0xFF. If MeasurementSummaryHash is 1, the combined measurement of the TCB is requested. This includes the Hardware Measurement, the SBB Fuse Measurement, and the Mini-Loader Measurement, as these are in the TCB for the generation of higher-level measurements. TCB_measurement = hash(HW_measurement_bitmap, hash(SBB_fuses), hash(MiniLoader)). If Measurement SummaryHash is 0xFF, then the response requires the overall measurement. ALL_measurement = hash(HW_measurement_bitmap, hash(SBB_fuses), hash(MiniLoader), hash(NVM_Bank)). In both cases, the measurements are inserted by the SBB as part of generating the response.
uint8 * S	ECDSA384 signature over the transcript hash. S is the size of the asymmetric signing algorithm output the responder selected in the last ALGORITHMS response message. The signature is done with the DeviceID private key and is populated by the SBB.
uint8 * 48	ResponderVerifyData This field shall equal the HMAC of the transcript hash, using finished_key as the secret key and using the negotiated hash algorithm as the hash function. In this device, this is a HMAC-SHA384 value.

10.10.5.9 Finish (0xE5)

Type	Request Data
uint8	SPDMVersion = 0x11
uint8	RequestResponseCode (FINISH = 0xE5)
uint8	Param1 Bit 0 - If set, the Signature field is included. This bit is set when mutual authentication occurs. All other bits reserved.
uint8	Param2 - Slot ID Only valid if Param1 = 0x01. Otherwise reserved. Slot number of the target Certificate Chain being authenticated in signature field. The value in this field is between 0 and 7 inclusive. It is 0xFF if the public key of the Responder was provisioned to the Requester previously.
uint8* 48	RequesterVerifyData This field is an HMAC of the transcript hash using the <i>finished_key</i> as the secret key and using the negotiated hash algorithm as the hash function. Since our device only supports SHA384, this will be an HMAC-SHA384.
Type	Response Data
uint8	SPDMVersion = 0x11
uint8	RequestResponseCode (FINISH_RSP = 0x65)
uint8	Param1 - Reserved
uint8	Param2 - Reserved
uint8*H	ResponderVerifyData

10.10.5.10 End Session (0xEC)

Type	Request Data
uint8	SPDMVersion = 0x11
uint8	RequestResponseCode (END_SESSION = 0xEC)
uint8	Param1 - See Appendix 10-85 .
uint8	Param2 - Reserved
Type	Response Data
uint8	SPDMVersion = 0x11
uint8	RequestResponseCode (END_SESSION_ACK = 0x6C)
uint8	Param1 - Reserved
uint8	Param2 - Reserved

Table 10-85. End Session Request Attributes

Offset	Field	Value	Description
0	Negotiated State Preservation Indicator	0	If the Responder supports Negotiated State caching (CACHE_CAP=1, the Responder preserves the Negotiated State. This is not supported on the E610.
		1	If the Responder supports Negotiated State caching, the Responder also clears the Negotiated State as part of session termination. This is not supported on the E610.
7:1	Reserved	Reserved	Reserved.

10.10.5.11 Respond If Ready (0xFF)

Type	Request Data
uint8	SPDMVersion = 0x11
uint8	RequestResponseCode (RESPOND_IF_READY = 0xFF)
uint8	Param1 - Original Request Code
uint8	Param2 - Token returned as part of <i>ResponseNotReady</i> extended error data/
Type	Response Data
...	Original command response.

10.10.6 Secure Manageability Support

When SPDm support is enabled, the device also supports secure manageability. The SPDm KEY_EXCHANGE command is used to create a secure session ID and a session key. Once the session is established, secured messages can be sent over MCTP as payload in the following format (the table below does not include the MCTP transport header). Note that all fields except for the MAC are in Little Endian. The Application Data is provided in its natural endianness.

Table 10-86. Secure Manageability Support

Payload Offset Byte	Bit	Field	Value and Description
0	7	IC	Shall be 0 for all type 6 messages.
0	6:0	Message Type	Always 6 in this case.
1-4	31:0	Session_ID	This is the Session ID established as part of the SPDm KEY_EXCHANGE. It is a concatenation of 2 bytes of a requester session ID and 2 bytes of a responder session ID chosen by both sides. Messages with a <i>Session_ID</i> which is not recognized by the device will result in error. Only a single session at any given time is supported over MCTP. This field is to be covered by the MAC.
5-6	15:0	Partial Sequence Number	Contains the lowest 16-bits of the sequence number. Note that the overall sequence number is 64-bits long and is tracked internally by the device, per session and per direction (that is, the device keeps track of a sequence number for messages it receives and a separate sequence number for messages it transmits). The first message sent in a session will be sent with a sequence number of zero. Each following message shall have a sequence number increased by one. Messages shall never be sent out of sequence. Messages with a lower-than-current sequence number shall be discarded. This field is to be covered by the MAC.
7-8	15:0	Length	Length of the secure packet data. The length shall be computed as a sum of the length of all following fields. This field is to be covered by the MAC.
9-10	15:0	Application Data Length	The length of data inside the encrypted payload. This field only describes the Application Data length and does not include the Random Data or following fields. This field is to be covered by the MAC and to be encrypted.
Variable	All	Application Data	The encapsulated payload (for example, an MCTP message). Note that type 6 messages are not allowed to be recursively encapsulated in type 6 messages. Such messages shall be discarded. This field is to be covered by the MAC and to be encrypted.
Variable	All	Random Data	Random data of random length. The size of the random data shall be between 1 byte and the maximum allowed by the underlying transport (minus the other fields described in this table). This field is to be covered by the MAC and to be encrypted.
Variable Start, 16 bytes	All	Message Authentication Code	The authentication tag output of AES-256-GCM.

10.11 Firmware Host Interface

10.11.1 Introduction

This chapter describes the interfaces available for software-to-firmware communication. There are two available channels:

1. A CSR based admin queue command for the use of driver & tools. There are 3 instances of this interface in this mode, that can be used by tools running on any of the PFs, as well as one driver interface on each PF.
2. Legacy CSR based host interface (as in Intel® Ethernet Controller X550)

In addition, a host isolate feature is available to allow the firmware to isolate the device from the host in case of a compromised host.

10.11.2 CSR-Based Firmware Admin Commands for Driver & Tools

Linkville implements a CSR based firmware admin commands interface for the use of the driver and tools. The CSR based firmware admin queue is implementing a single command interface to firmware which is CSR-based. Linkville implements a memory space including Sync Response (32 byte descriptor), Async Response (32 byte descriptor) and Event Response Buffer (4K bytes buffer). This area would be directly accessible by the user mode tools software.

The GL_HICR and GL_HICR_EN registers are used to control the usage of this interface. In addition the GL_SWSM register is used as a semaphore, to allow the tools to share this interface.

10.11.2.1 CSR-Based Firmware Commands Interface Hardware Implementation

The tools software would write a single admin command, using the format shown in the table below. Then it will signal the firmware using a GL_HICR CSR, as defined below that it wrote command for execution. The firmware will read the command, execute it and will use the same memory space to send back the response from the admin command.

Note: No asynchronous events can be used by the tool queue (e.g. link change event), except asynchronous events that are caused by delayed response to a command.

The CSRs of this interface are accessible to all PFs, but assumption is that only PF0 will access it. The tool can run from any function, but single instance of the tool should talk to the hardware and the GL_SWSM hardware semaphore can be used to control that. The CSR based firmware admin queue implements:

- 64 bytes of descriptor area
- 4K bytes of buffer area

Those areas are accessible for software reads and writes only if GL_HICR.EN bit is set. Write access is also limited by GL_HICR.C being zero.

Those areas are also mapped as Aux registers – though aux, the memory is always accessible.

Table 10-87. CSR-Based Firmware Admin Commands Areas

Area	CSR Name	Size	Offset CSR	Usage in Command	Usage in Response
Data Buffer	GL_HIBA	4K Bytes	0x81000	Command Buffer	Event Response Buffer
1st Descriptor	GL_HIDA	32 Bytes	0x82000	Command Descriptor	Sync response
2nd Descriptor	GL_HIDA+0x20	32 bytes	0x82020	N/A	Async response
Data Buffer	PF_HIBA	4K Bytes	0x84000	Command Buffer	Event Response Buffer
1st Descriptor	PF_HIDA	32 Bytes	0x85000	Command Descriptor	Sync response
2nd Descriptor	PF_HIDA+0x20	32 Bytes	0x85020	N/A	Async response

The CSR based firmware admin queue also implements the following registers:

GL_HICR and GL_HICR_EN. The semaphore is implemented with the GL_SWSM register.

Table 10-88. GL_HICR

Bits	Field	Init Value	FW (Aux) Access	SW (CSR) Access	Description
0	Reserved	0b	RW	RW	Reserved.
1	C	0b	RW1C	RW (1)	Command The tool sets this bit when it has finished putting a command block in the command buffer This bit should be cleared by the firmware when the command's processing is completed. Setting this bit causes an interrupt to the firmware. This bit can be only set through the CSR (software) interface and can be cleared through the AUX (firmware) interface While this bit is set the memory is RO to software.
2	SV	0b	RW	RW	Status Valid Indicates that there is a valid status in the 1st descriptor that the device driver can read. If Flags.BUF is set in the 1st descriptor, the buffer is valid 0b = Status not valid. 1b = Status valid.
3	EV	0b	RW	RW	Event Valid Indicates that there is a valid status in the 2nd descriptor that the device driver can read. If Flags.BUF is set in the 2nd descriptor, the buffer is valid 0b = Status not valid. 1b = Status valid.

Changing the C bit from 0 to 1 in this CSR would create an interrupt to firmware. The firmware will implement an interrupt routine, which reads the admin queue command from the internal memory and execute it, in a similar way as it executes any other admin command. The results of the admin command, would be written back to the same memory.

Table 10-89. GL_HICR_EN

Bits	Field	Init Value	FW (Aux) Access	SW (CSR) Access	Description
0	EN	0b	RW	RO	Enable When set, indicates that the buffer is accessible for the device. Note: This bit is common to all functions.

Table 10-90. GL_SWSM

Bits	Field	Init Value	FW (Aux) Access	SW (CSR) Access	Description
31:0	SMBO	0xFFFF_FFFF	RW	RW	Semaphore Owner Software should write its PF# value as read from the PF_FUNC_RID HIF register to try and get the semaphore. The hardware will set this value only if the current value is 0xFFFF_FFFF. The software should then read it again to see if the semaphore was obtained. The software releases the semaphore by writing 0xFFFF_FFFF When a software obtained or released the semaphore, the firmware gets an interrupt to allow timeout implementation. firmware will set this field to 0xFFFF_FFFF if the node owning the semaphore was reset (PERST#). The node can be read from bits 18:16 of this register. It should also reset it after a timeout.

10.11.2.2 Firmware and Software Interaction Using the CSR-Based Admin Commands

The following interaction is expected:

1. Firmware need to register to an interrupt, which is triggered when software is writing to GL_HICR. The mechanism has to be enabled by setting GL_HICR.EN bit (GL_HICR.EN = 1)
2. Software fills the descriptor + buffer, sets GL_HICR.C and clears GL_HICR.SV and GL_HICR.EV.
3. Once the GL_HICR.C bit is set, the firmware on the main CPU will read the data (descriptor + buffer), parse the packet, and check the command parameters correctness.

Note: All the commands reach the main CPU – and if needed, it should distribute them to other CPUs.
4. Firmware executes the command (arbitration with other commands is left to firmware).
5. The firmware then writes a response in the 1st descriptor and optionally to the buffer and sets the SV bit. If there is an error, or the command is completed (synchronous command), it also clears the GL_HICR.C bit
6. If Flags.BUF is set in the 1st descriptor, the buffer is valid and the software should process it.
7. If GL_HICR.C bit is cleared, software stops the flow.
8. If the command is asynchronous, firmware waits for the command to complete. It does not process additional commands.
9. Firmware responds using the same 4KB buffer and the 32B 2nd descriptor, clears the GL_HICR.C bit and sets the GL_HICR.EV field.
10. If Flags.BUF is set in the 2nd descriptor, the buffer is valid and the software should process it.

Note: The buffer may be used for the 1st or 2nd response but not for both.

Note: The mechanism mimics the synchronous and asynchronous flows used in regular event queues, but the command buffer and completion buffer are shared. The tool should not post new commands until the C bit is cleared.

Note: When C bit is set, the entire memory (buffer and descriptors) is RO to software.

10.11.2.3 Admin Commands Descriptor Formats

The Admin Commands uses the following descriptor structure. All descriptors and commands are defined using Little endian notation with 32 bit words. Drivers using other conventions should take care to do the proper conversions.

Table 10-91. Control Commands Descriptor Structure (in LE 32 Order)

+3								+2								+1								+0													
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0						
Opcode																F	E	I	S	B	V	R	L						V	E	E	R	C	M	P	D	D
Return Value																Data Len																					
Cookie High																																					
Cookie Low																																					
Param0																																					
Param1																																					
Data address high																																					
Data address Low																																					

Table 10-92. Control Descriptor Field Descriptions

Name	Bytes.Bits	Description
Flags.DD	0.0	Set by firmware to mark entry done
Flags.CMP	0.1	Set by firmware to mark entry as completion
Flags.ERR	0.2	Set by firmware to mark entry as an error indication
Flags.VFE	0.3	Set by firmware to mark entry as an event forwarded from a VF driver
Flags.Reserved	0.4-1.0	Reserved, must be zeroed by sender and ignored by receiver.
Flags.LB	1.1	Set by driver to indicate that indirect buffer is longer than AQ_LARGE_BUF
Flags.RD	1.2	Set by driver to indicate that firmware needs to read indirect buffer.
Flags.VFC	1.3	Set by driver to indicate command on behalf of a VF.
Flags.BUF	1.4	This command uses additional data
Flags.SI	1.5	Do not interrupt when this command completes
Flags.EI	1.6	Interrupt on error - supersedes Flags.SI in case of an error
Flags.FE	1.7	If previous command completed in error, flush this one
Opcode	2-3	Command opcode
Datalen	4-5	indirect data length in bytes (can be used for other uses if Flags.BUF is unset)
Return value	6-7	Return value for command or event.It is used by W as a return value and reflects the error code as described in Section 10.11.2.5 and Table 10-98 .
Cookie High	8-11	Opaque data, echoed by receiver, high half
Cookie Low	12-15	Opaque data, echoed by receiver, Low half
Param0	16-19	First general use parameter
Param1	20-23	Second general use parameter

Table 10-92. Control Descriptor Field Descriptions [continued]

Name	Bytes.Bits	Description
Data Address high	24-27	indirect data pointer (can be used for other uses if Flags.BUF is unset)
Data Address low	28-31	indirect data pointer (can be used for other uses if Flags.BUF is unset)

10.11.2.4 Command Descriptions

10.11.2.4.1 Direct Command

10.11.2.4.1.1 Direct Admin Command

The template for a command that is fully contained in the descriptor and does not need an additional data buffer.

Table 10-93. Direct Admin Command Structure

Name	Bytes.Bits	Value	Remarks
Flags.DD	0.0	0	Driver must clear
Flags.CMP	0.1	0	Driver must clear
Flags.ERR	0.2	0	Driver must clear
Flags.VFE	0.3	0	Driver must clear
Flags.Reserved	0.4-1.0	0	Reserved, must be zeroed by sender and ignored by receiver.
Flags.LB	1.1	0	A direct command has no additional buffer
Flags.RD	1.2	0	A direct command has no additional buffer
Flags.VFC	1.3	0	Driver must clear
Flags.BUF	1.4	0	A direct command does not have an additional write buffer
Flags.SI	1.5	Driver May set	Do not interrupt when this command completes
Flags.EI	1.6	Driver May set	Interrupt on error - supersedes Flags.SI in case of error
Flags.FE	1.7	Driver May set	If set, command will be flushed if the preceding command resulted in an error
Opcode	2-3	Opcode	Command opcode
Datalen	4-5	0	
Return value	6-7		
Cookie High	8-11	Cookie	Opaque value, will be copied by the firmware into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the firmware into the completion of this command
Param0	16-19		First command parameter
Param1	20-23		Second command parameter
Data Address high	24-27		Can be used for an additional command parameter
Data Address low	28-31		Can be used for an additional command parameter

10.11.2.4.1.2 Direct Command Completion

Table 10-94. Direct Command Completion Event Template

Name	Bytes.Bits	Value	Remarks
Flags.DD	0.0	1	Firmware must set
Flags.CMP	0.1	1	Firmware must set
Flags.ERR	0.2	0 or 1	Firmware must set only if it reporting an error
Flags.VFE	0.3	0	Firmware must clear
Flags.Reserved	0.4 -1.0	0	Reserved, must be zeroed by sender and ignored by receiver.
Flags.LB	1.1	echo	Firmware will copy value from command
Flags.RD	1.2	echo	Firmware will copy value from command
Flags.VFC	1.3	echo	Firmware will copy value from command
Flags.BUF	1.4	echo	Firmware will copy value from command
Flags.SI	1.5	echo	Firmware will copy value from command
Flags.EI	1.6	echo	Firmware will copy value from command
Flags.FE	1.7	echo	Firmware will copy value from command
Opcode	2-3	Opcode	Command opcode
Datalen	4-5		Can be used for an additional command parameter
Return value / VFID	6-7		Firmware return value 0=no error (for error codes see Table 10-98).
Cookie High	8-11	echo	Opaque value, will be copied by the firmware from the command
Cookie Low	12-15	echo	Opaque value, will be copied by the firmware from the command
Param0	16-19		First command parameter
Param1	20-23		Second command parameter
Data Address high	24-27		Can be used for an additional command parameter
Data Address low	28-31		Can be used for an additional command parameter

10.11.2.4.2 Indirect Command

10.11.2.4.2.1 Indirect Admin Command

An indirect write command uses the buffer specified in the descriptor

The BUF flag must be set by the driver. If the buffer is larger than 512 bytes the LB flag must be set.

This version of the commands is limited to buffers up to 4096 bytes. If the command uses the buffer to pass data the RD flag needs to be set.

Table 10-95. Indirect Command Template

Name	Bytes.Bits	Value	Remarks
Flags.DD	0.0	0	Driver must clear
Flags.CMP	0.1	0	Driver must clear
Flags.ERR	0.2	0	Driver must clear
Flags.VFE	0.3	0	Driver must clear
Flags.Reserved	0.4-1.0	0	Reserved, must be zeroed by sender and ignored by receiver.
Flags.LB	1.1	Driver May set	Set by driver if buffer is longer than AQ_LARGE_BUF (512)
Flags.RD	1.2	Driver May set	Set by driver to indicate that firmware needs to read indirect buffer.
Flags.VFC	1.3	0	Driver must clear
Flags.BUF	1.4	1	This command uses additional data buffer. Driver must set this flag on an indirect command.
Flags.SI	1.5	Driver May set	Do not interrupt when this command completes
Flags.EI	1.6	Driver May set	Interrupt on error - supersedes Flags.SI in case of error
Flags.FE	1.7	Driver May set	If set, command will be flushed if the preceding command resulted in an error
Opcode	2-3	Opcode	Command opcode
Datalen	4-5	Buffer length	Usable length of additional buffer in bytes
Return value	6-7		
Cookie High	8-11	Cookie	Opaque value, will be copied by the firmware into the completion of this command
Cookie Low	12-15	Cookie	Opaque value, will be copied by the firmware into the completion of this command
Param0	16-19		First command parameter
Param1	20-23		Second command parameter
Data Address high	24-27	0	Reserved (not used by CSR based interface)
Data Address low	28-31	0	Reserved (not used by CSR based interface)

10.11.2.4.2.2 Indirect Command Completion

When completing an indirect command the firmware will overwrite the *datalen* with the actual length of data returned by the command.

Table 10-96. Indirect Command Completion Template

Name	Bytes.Bits	Value	Remarks
Flags.DD	0.0	1	Firmware must set
Flags.CMP	0.1	1	Firmware must set
Flags.ERR	0.2	0 or 1	Firmware must set only if it reporting an error
Flags.VFE	0.3	0	Firmware must clear
Flags.Reserved	0.4 -1.0	0	Reserved, must be zeroed by sender and ignored by receiver.
Flags.LB	1.1	echo	Firmware will copy value from command
Flags.RD	1.2	echo	Firmware will copy value from command
Flags.VFC	1.3	echo	Firmware will copy value from command
Flags.BUF	1.4	echo	Firmware will copy value from command
Flags.SI	1.5	echo	Firmware will copy value from command
Flags.EI	1.6	echo	Firmware will copy value from command
Flags.FE	1.7	echo	Firmware will copy value from command
Opcode	2-3	Opcode	Command opcode
Datalen	4-5	echo	Firmware will copy value from command
Return value / VFID / PFID / DRID	6-7		Firmware return value 0=no error (for error codes see Table 10-98)
Cookie High	8-11	echo	Opaque value, will be copied by the firmware from the command
Cookie Low	12-15	echo	Opaque value, will be copied by the firmware from the command
Param0	16-19		First command parameter
Param1	20-23		Second command parameter
Data Address high	24-27		Can be used for an additional command parameter
Data Address low	28-31		Can be used for an additional command parameter

10.11.2.5 Error Handling

When firmware encounters an error it uses the return value field to indicate the type of error. The error code is comprised of two bytes, the lower byte is a user-visible code from [Table 10-98](#) the higher byte may be used by firmware to report internal state or debug information, the driver must log this information, firmware may change this value from release to release. It is not to be reported to the user and no other action is to be taken upon this data. when the return value is 0 ("no error") firmware must not set the high byte.

Table 10-97. Control Commands Return Value Fields

Name	Bytes	Comment
Code	0:7	Return value from Table 10-98
FW Internal Code	8:15	Firmware internal code, must be 0 if Code field is 0

10.11.2.6 Error Codes

When firmware completes a command it uses the following error codes.

Table 10-98. Control Commands Return Values and Error Codes

Error Code	Value	Meaning
(No Error)	0	No Error (success)
EPERM	1	Operation not permitted
ENOENT	2	No such element
ESRCH	3	Bad opcode
EINTR	4	operation interrupted
EIO	5	I/O error or firmware internal error
ENXIO	6	No such resource
E2BIG	7	Arg too long
EAGAIN	8	Try again
ENOMEM	9	Out of memory
EACCES	10	Permission denied
EFAULT	11	Bad address
EBUSY	12	Device or resource busy
EEXIST	13	Attempt to create something that exists
EINVAL	14	Invalid argument
ENOTTY	15	Not a typewriter
ENOSPC	16	No space left or allocation failure
ENOSYS	17	Function not implemented
ERANGE	18	Parameter out of range
EFLUSHED	19	Command flushed because a previous command completed in error
BAD_ADDR	20	Internal error, descriptor contains a bad pointer
EMODE	21	Operation not allowed in current device mode
EFBIG	22	File Too Big

10.11.2.7 Admin Commands

Table 10-99. Generic Commands

Name	Opcode	Type	Section Reference
Get Version	0x0001	Direct	10.11.2.7.1
Driver Version	0x0002	Indirect	10.11.2.7.2
Reserved	0x0003	Indirect	
Set PF context	0x0004	Direct	10.11.2.7.3
Reserved	0x0005	Direct	
Request Resource Ownership	0x0008	Direct	10.11.2.7.3
Release Resource Ownership	0x0009	Direct	10.11.2.7.5
Discover Function Capabilities	0x000A	Indirect	10.11.2.7.6
Discover Device Capabilities	0x000B	Indirect	
Disable RXEN	0x000C	Direct	10.11.3.3.2
Get Firmware Event Command	0x0014	Direct	10.11.2.7.9
Reserved	0x0C31	Indirect	
Reserved	0X0C90	Direct	
Disable RX		Direct	

Table 10-100. Manufacturing Commands

Name	Opcode	Type	Section Reference
Program Fuse Bits	0xDD00	Indirect	10.11.2.7.10
Read Fuse Bits	0xDD01	Indirect	10.11.2.7.11
Reserved	0xDD02	Indirect	
Reserved	0xDD03	Direct	
Reserved	0xDD04	Direct	

10.11.2.7.1 Get Version (0x0001)

This must be the first command that the driver issues before it can use the queue for other purposes. The driver must inspect the reply to ensure that the firmware version is compatible. If firmware is still initializing it may delay response until it is done. Both the firmware and the API have two unassigned 16-bit values as minor and major version. The driver must not continue loading if the major version mismatch. Minor versions are for tracking changes that do not need driver modifications.

Table 10-101. Get Version Command

Name	Bytes.Bits	Value	Remarks
Flags	1-0		See Section 10.11.2.4.1.1 for details.
Opcode	2-3	0x0001	Command opcode.
Datalen	4-5	0x0	no external response buffer.
Return value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
ROM Build ID	16-19		Device ROM Build version.
FW Build ID	20-23		Device firmware build version.
FW Branch	24		Firmware Branch Identifier (unsigned 8-bit integer).
FW Major Version	25		Firmware major version (unsigned 8-bit integer).
FW Minor Version	26		Firmware minor version (unsigned 8-bit integer).
FW Patch Version	27		Firmware patch version (unsigned 8-bit integer).
AQ API Branch	28		AQ API Branch Identifier (unsigned 8-bit integer).
AQ API Major Version	29		AQ API major version (unsigned 8-bit integer).
AQ API Minor Version	30		AQ API minor version (unsigned 8-bit integer).
AQ API Patch Version	31		AQ API patch version (unsigned 8-bit integer).

10.11.2.7.2 Driver Version (0x0002)

This command is used by the driver to report the Driver version. The firmware should use the driver version in order to report it to the BMC.

Table 10-102. Driver Version Command

Name	Bytes.Bits	Value	Remarks
Flags	1-0		See Section 10.11.2.4.2.1 for details.
Opcode	2-3	0x0002	Command opcode.
Datalen	4-5		Buffer Length. Can be up to 32.
Return value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
Driver Version	16-19	version	Byte 16 = major version Byte 17 = minor version Byte 18 = build version Byte 19 = sub-build version
Reserved	20-23	0x0	Reserved.
Data Address High	24-27		Address of response buffer.
Data Address Low	28-31		Address of response buffer.

Table 10-103. Driver Version Buffer

Name	Length	Description
Driver Version	Datalen	Driver string (not null terminated) as reported by driver:

10.11.2.7.3 Set PF Context (0x0004)

This admin command is used to set explicit PF ID number. It is needed for some admin commands that requires control on specific PF context regardless of the PF from which the admin command is initiated.

Table 10-104. Set PF Context Command

Name	Bytes.Bits	Value	Remarks
Flags	1-0		See Section 10.11.2.4.2.1 for details.
Opcode	2-3	0x0004	Command opcode.
Datalen	4-5		No external data
Return value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
PF ID	16		Physical function ID.
Reserved	17:31	0x0	Reserved.

10.11.2.7.4 Request Resource Ownership (0x0008)

This command is used by the driver to request ownership of a shared resource. The driver specifies the resource and the type of access it requests (listed in [Table 10-105](#)). On success the command returns a return value of 0. The completion specifies the maximum time in ms that the driver may hold the resource in the Timeout field. The driver must free the resource before that time. The driver must free a resource before asking for it again. A request for a resource that is already held by this driver will fail with the EACCESS error code. A timeout value of zero means no timeout.

If the resource is held by someone else and the resource is different from “Global Config Lock”, the command completes with a EBUSY return value and the timeout field indicates the maximum time the current owner of the resource has to free it. For the “Global Config Lock” resource, the command always completes successfully, while the status is reflected in the *Status* field of the command.

Firmware implements a timeout mechanism taking back the ownership if the driver hangs. Any further commands by this driver that attempt to access this resource will fail with the EPERM error code until the driver frees the resource and requests it again.

Table 10-105. Shared Resources

Resource	ID	Supported Modes	Default Timeout
NVM	0x0001	1=read, 2=write	3000 ms for read, 180000 ms for write

Table 10-106. Request Resource Ownership Command

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section 10.11.2.4.1.1 for details.
Opcode	2-3	0x0008	Command opcode.
Datalen	4-5	0x0	No external buffer for this command.
Return value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
Resource ID	16-17	ID	See Table 10-105 for list of IDs.
Access Type	18-19	Type	See Table 10-105 for list.
Timeout	20-23	timeout	Timeout in ms. For specific resources, such as Change Lock - 0x0003 and Global config Lock - 0x0004, this field is used by software to override the default timeout for the operation and also written by firmware to specify the timeout used for this operation. As an input: the software might specify timeout longer than the default taken for this resource and up to one minute. As an output: This field is returned in the firmware response structure and indicates the timeout used for the specific resource.
Resource Number	24-27	Number	For an SDP, this is the pin ID of the SDP.
Status	28-29	Status	Written by firmware. Used for Global Config Lock (Resource ID 0x0004), reserved for others. 0x0000 = Global Config Lock acquired successfully (STAT_SUCCESS). 0x0001 = Global Config Lock acquired by another driver, configuration is in progress (STAT_INPROGRESS). 0x0002 = Global Lock could not be acquired, configuration was already completed (STAT_COMPLETED).
Reserved	30-31	0x0	Reserved.

Notes:

- The rest of the drivers may use Request Resource Ownership with Resource ID = 0x0004 admin command only, until getting STAT_COMPLETED status.
- Global Config Lock could be successfully acquired only once after Master/MAC reset or after the last driver goes down, to the first driver asking for the lock for the first time.
- The "Request Resource ownership" admin command might return additional error code. EACCES_BMCU (0x24) - BMC update in progress - Returned when NVM ownership is required during PLDM firmware update.

10.11.2.7.5 Release Resource Ownership (0x0009)

This command is used to return ownership of a shared resource to the firmware. The driver will specify the ID of the resource it is releasing in the ID field.

Table 10-107. Release Resource Ownership Command

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section 10.11.2.4.1.1 for details.
Opcode	2-3	0x0009	Command opcode.
Datalen	4-5	0x0	No external buffer for this command.
Return value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
Resource ID	16-17	ID	see Table 10-105 for list of IDs.
Reserved	18-23	Reserved	Reserved.
Resource Number	24-27	Number	if a resource number was specified in the request, it needs to be specified here too.
Reserved	28-31	0x0	Reserved.

10.11.2.7.6 Discover Device/Function Capabilities (0x000A/(0x000B))

This command is used to request the list of capabilities of the device or the function. The firmware will fill in the capabilities structure and return the length to the driver. If the buffer size is not big enough for the whole structure, the firmware will return ENOMEM (at the return value), write the number of capabilities that should be returned by the command to the cap count field and reset the length field to zero. This can be used to discover the number of capabilities returned and the structure size can be calculated by software using the known size of the capability structure. Capabilities are described using the structure in [Table 10-109](#) the list of resources recognized by this version of the command are in [Table 10-110](#). Additional capabilities can be retrieved using the Get PHY Abilities (0x0600) command.

Note: Unsupported capabilities are not reported.

Table 10-108. Discover Device/Function Capabilities Commands

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section 10.11.2.4.2.1 for details.
Opcode	2-3	0x000A or 0x000B	Command opcode 0x000A for function capabilities 0x000B for device.
Length	4-5	Buffer length	Length of buffer.
Return value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16-19	0x0	
Cap Count	20-23		Number of capability records returned (zeroed by driver written by firmware).
Data Address High	24-27	0x0	Address of response buffer.
Data Address Low	28-31	0x0	Address of response buffer.

Table 10-109. Capability Structure

Name	Length	Remarks
Capability	16 bits	See Table 10-110 for list of capabilities.
Major Version	8 bits	
Minor Version	8 bits	
Number	32 bits	Number of resources described by this capability.
Logical ID	32 bits	Only meaningful for some types of resources.
Physical ID	32 bits	Only meaningful for some types of resources.
Additional data1	64 bits	In the E610 this field is used in the "Logical to Physical port mapping" (0x73) capability structure.
Additional data2	64 bits	In the E610 this field is used in the "Logical to Physical port mapping" (0x73) capability structure.

Table 10-110. Resources Recognized by This Version of the Command

Name	Capability	Ver	Number	Logical ID	Physical ID
Reserved	0x0001	1.0		N/A	
Manageability Mode	0x0002	2.0	Bits 3:0 Returns the value of the "Manageability Pass Through Mode" field in the NVM Bits 7:4: Returns the value of the "Control Interface" field in the NVM Bits 11:8: Returns the value of the "Redirection Sideband Interface" field in the NVM Bits 31:12: Reserved	Supported Protocols over MCTP Bit 0: Reserved Bit 1: PLDM Bit 2: OEM commands Bit 3: NC-SI Bit 31:4: Reserved	Supported Protocol over PLDM (relevant only in PLDM is supported): Bit 0: PLDM base (always set if PLDM supported) Bit 1: Reserved Bit 2: PLDM for Platform Monitoring and Control Bit 3:4 Reserved Bit 5: PLDM firmware update Bit 6: Reserved (RDE)
OS2BMC Capable	0x0004	1.0	Returns the value of the "OS2BMC Capable" field in the NVM		
Functions Valid	0x0005	1.0	Bits 1:0 each corresponds to a PFPCI_STATUS1.FUNC_VALID bit for PFs 0,1		
Alternate RAM Structure	0x0006	1.0	1		
WoL	0x0008	1.0	Number of ACPI supported filters = 8 if supported, 0 otherwise. In the E610 it is equal to 0, excluding Magic Packet filters which are always supported. Same value for all ports and for the device command. For the device return the total number of filters across all supporting functions.	SEID of WoL VSI of the port. N/A if programming method is HW. N/A for device.	Bit0: APM WoL is supported (based on PFPM_APM.APME bit For Device command - OR of all the PFs bits. Bit 1: ACPI Programming Method (relevant only if Number of ACPI supported filters is not zero) 0b = HW 1b = AQC For Device command - OR of all the PFs bits. Other bits - reserved
SR-IOV	0x0012	1.0	Set to one if enabled in config space PCI_CNF2.NUM_VFS>0PCI_CAPSU P For device should be set if set in any of the functions.	SR-IOV version (1.1)	
Virtual Function	0x0013	1.0	Function: 64 Device: Total number of VFs exposed to all functions128	Logical ID of first VF 0x0	
VMDq	0x0014	1.0	Set to one		
802.1Qbg	0x0015	1.0	One if enabled		

Table 10-110. Resources Recognized by This Version of the Command [continued]

Name	Capability	Ver	Number	Logical ID	Physical ID
VSI	0x0017	1.0	Function: Number of guaranteed VSI As read from PF allocations structure for PFs Device: Number of VSIs allocated to the host (not including EMP VSIs).		
DCB	0x0018	1.0	One if enabled	Device = 0: Bitmap of active TCs	Max number of TCs (8 TCs when the device is configured to 1, 2 or 4 ports, 4 TCs when the device is configured to more than 4 port)
Reserved	0x0021	1.0	Reserved 0x0		
RSS	0x0040	1.0	Table size 2048 for PFs. 64 for VFs.	Entry width in bits 8 for PFs. 4 for VFs.	
RX Queues	0x0041	1.0	Function: Number of queues allocated to the PF. Device: Total number of queues available to the device		ID of first queue
TX Queues	0x0042	1.0	Function: Number of queues Device: Total number of queues available.		ID of first queue
MSI-X	0x0043	1.1	Number of vectors For functions: PFINT_ALLOC[PF#].LAST - PFINT_ALLOC[PF#].FIRST + 1 For the device: 2048		First allocated vector: PFINT_ALLOC[PF#].FIRST
VF-MSIX ¹	0x0044	1.0	Number of MSIX Vectors available to VFs of this PF		
Flow Director	0x0045	1.0	Function: Number of filters guaranteed to this PF. Device: Number of filters available in device (8192)	Function: Number of best effort filters Device: Number of filters available in device (8192)	

Table 10-110. Resources Recognized by This Version of the Command [continued]

Name	Capability	Ver	Number	Logical ID	Physical ID
1588	0x0046	1.1	Function: Byte 0 (LSB) - timer information: bit 0 - TimeSync is enabled on the port on which this function runs bit 1 - a master timer is owned by this function. bit 2 - timer enabled (Taken from NVM structure "1588 functionality enablement"). bit 3 - reserved. bit 4 - timer index owned by the function (0/1). (Note: This bit defaults to 0, in the case that the function owns both timers). bits 7:5 - reserved Byte 1 - GPIOs association with the timer owned by the function: bit 8 - SDP_TIMESYNC[0] bit 9- SDP_TIMESYNC[1] bit 10 - SDP_TIMESYNC[2] bit 11 - SDP_TIMESYNC[3] bit 12 - 1PPS (output) bit 13 - TIME_SYNC (input) bit 14 - CLK_SYNC (output) bit 15 - reserved. Byte 2 - Clock source info: bits 18:16 - clock frequency report from CGU. bit 19 - reserved. bit 20 - clock source selection report from CGU. bits 23:21 - reserved. Byte 3 (MSB) - PHY association with timers: bit 24 - timer index associated with the port on which the function runs. bit 31:25 - reserved. Device: Byte 0 (LSB) - timer ownership: bits 2:0 - PF owning timer 0. bit 3- Timer 0 is owned by a PF. bits 6:4 - PF owning timer 1. bit 7- Timer 1 is owned by a PF. Byte 1 - GPIOs associated with any of the timers (same structure as Byte 1 in function response). Byte 2 - 1588 Clock source info (same as Byte 2 in function response). Byte 3 - General information: bit 24 - TimeSync enabled on any of the ports bit 25 - Timer 0 is enabled. bit 26 - Timer 1 is enabled. bit 31:27 - reserved.	Function: N/A Device: A bitmap of the enabled ports	Function: N/A Device: Timer ownership bitmap of the enabled ports (each bit value shall hold the timer index for the port).
MaxMTU	0x47	1.0	For function: Max MTU of the function. For device: Max MTU of the hardware (9728)		

Table 10-110. Resources Recognized by This Version of the Command [continued]

Name	Capability	Ver	Number	Logical ID	Physical ID
NVM versions ²	0x48	1.0	1 st word – NVM word address, 2 nd word – NVM value	1 st word – NVM word address, 2 nd word – NVM value	1 st word – NVM word address, 2 nd word – NVM value
NVM pending versions ³	0x49	1.3	1 st word – NVM word address, 2 nd word – NVM value	1 st word – NVM word address, 2 nd word – NVM value	1 st word – NVM word address, 2 nd word – NVM value
OROM version	0x4A	1.3	2 words - OROM version		
OROM pending version ⁴	0x4B	1.3	2 words - OROM version		
Netlist version ⁵	0x4C	1.3	BaseReleaseVersion.Major	BaseReleaseVersion.Type	BaseReleaseVersion.IANA + CustomerNetlistVersion
Netlist pending version ⁶	0x4D	1.3	BaseReleaseVersion.Major	BaseReleaseVersion.Type	BaseReleaseVersion.IANA + CustomerNetlistVersion
Reserved	0x0051	1.0			
LED ⁷	0x0061	1.0	Always 1	Action	pin number (GPIO Index)
SDP ⁸	0x0062	1.0	Always 1	Action	pin number (GPIO Index)
MDIO ⁹	0x0063	1.0	One if enabled		The interface used to control this port.
Drop/No Drop policy ¹⁰	0x0065	1.0	Returns the value of the PXE Mode No-Drop Policy supported NVM bit.	Function: 0x1 if No-Drop policy is enabled on the port used by the current PF. 0x0 if No-Drop policy is not enabled on the port used by the current PF. Device: N/A	
HII Port Disable Allowed	0x0066	1.0	For device and Function returns the value of "Features enable: Allow HII Port Disable" bit from NVM. For Function 0 returns always 0 (no supported for PF0).		
Reserved	0x0071	1.0	Reserved	Reserved	Reserved
Reserved	0x0072	1.0		Enabled ports bitmap. A bit map [19:0], with 1 set for each enabled logical port.	
SKU ¹¹	0x0074	1.0	Returns the SKU as described in register GL_UFUSE_SOC[15:0]	Reserved	Reserved
Port Mapping ¹²	0x0075	1.3	Reserved	Function Number	Port Number
PCIe reset avoidance supported version	0x0076	1.0	1 - pre version 2 - full support		

Table 10-110. Resources Recognized by This Version of the Command [continued]

Name	Capability	Ver	Number	Logical ID	Physical ID
Post Update Double Reset	0x077	1.0	Post Update Double Reset solution 0 - Not Supported 1 - Supported Note: when Capability is 0 and solution is not supported, it is recommended to not use EMPR prior to reboot as PCI may become unresponsive.		
NVM Mgmt ¹³	0x0080	1.0	Bitmask: Bit 0: SRev Disabled (0 = Do not allow rollback; 1 = Allow rollback) Bit 1: Update Disabled (0 = no lockdown; 1 = lockdown) Bit 2: Lockdown supported 0 = not supported; 1 = supported. Bit 3: PFA modification via firmware (DCR153) support: 0 = no support; 1 = support Bit 4: DCR161 supported		
Reserved	0x0081	3.1			
Reserved	0x0082	3.1			
Reserved	0x0083	3.1			
Reserved	0x0084	3.1			

1. This capability is not available in the E610, as any vector can be assigned to a VF, and each VF can get a different number of vectors
2. The NVM Versions capability is used to return NVM version data to software in a centralized location. It will appear multiple times, reporting three NVM versions per entry. The following NVM words are returned as versions: 0x18-0x2E (inclusive) and 0x34, 0x35. An NVM word address of 0xFFFF indicates an invalid field.
3. This capability takes the same format as in NVM versions capability. This capability appears only if pending image exists. This capability is exposed in recovery mode.
4. This capability appears only if pending image exists.
5. This capability is exposed in recovery mode.
6. This capability appears only if pending image exists. This capability is exposed in recovery mode.
7. Repeat this entry for each assigned LED. In E610, LEDs are not assigned to functions and therefore this capability is not returned if called at the context of a function. If Device = 1, returns an entry for each LED in the device, in E610 returns an entry for each GPIO in the IO widget, which is defined as LED in the link topology.
8. Repeat this entry for each assigned pin. In E610, SDPs are not assigned to functions and therefore this capability is not returned if called at the context of a function. If Device = 1, returns an entry for each SDP in the device, in E610 returns an entry for each GPIO in the IO widget, which is defined as SDP in the link topology.
9. This entry is not relevant for E610 and is not returned.
10. The values in this capability are based on the PXE Mode "No-Drop Policy Supported" NVM bit and on variable sw_force_no_drop.
11. This capability is exposed in recovery mode.
12. For Discover Device Capabilities this entry is returned for each enabled function. For Discover Function Capabilities it is returned only for the current function
13. This capability is exposed in recovery mode.

Note: Fields that are not applicable (empty in the table) are set to zero by the firmware.

Note: When device capabilities are requested total numbers for the whole device should be returned.

10.11.2.7.6.1 Software Activities Prior to Calling VM/VF Reset

- Software must stop sending doorbells to the disabled queues before closing them.
- For TX queues which are associated with doorbell queue, the Doorbell queue must be drained from any doorbell message of the disabled queues.
 - This is done by sending a draining marker through the doorbell queue. A Doorbell queue descriptor with Dummy and RS bits set acts as a drain marker.
- Stop getting interrupts for the disabled queue(s)
 1. Software should clear the cause_ena bit in the QINT_TQCTL register for all Disabled queues.
 2. Software waits 100 ns.
 3. Software sends software interrupt for the vector associated to the queue.
 4. When interrupt arrives, software can continue...
- In case of VM reset or multiple queues disable, software will do step #a for all disabled queues and step #c for all vectors associated to the queues.

10.11.2.7.6.2 Software Activities After VM/VF Reset AQ Completed

- Once the whole operation is complete and the pipe is cleaned from the disabled queues (EMP firmware responses with "No Error", software may re-use all involved resources.
- Software may post a VM reset notification via the completion queue. This is done directly to each involved completion queue using CSRs GLCOMM_CQ_CTL[512].

10.11.2.7.7 Clear PXE Mode (0x0110)

The Clear PXE Mode Admin Command transits the device from PXE to non-PXE mode. The structure of the admin command and its completion are shown below.

Table 10-111. Clear PXE Mode Command

Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Section 10.11.2.4.2.1 for details.
Opcode	2-3	0x0110	Command opcodes.
Datalen	4-5	0x0	N/A (reserved zero).
Return value/VFID	6-7	0x0	N/A (reserved zero).
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16	0x2	Reserved 0x2
Reserved	17-31	0x0	

Table 10-112. Clear PXE Mode Admin Command Completion

Name	Bytes.Bits	Value	Remarks
Flags	0-1		See Section 10.11.2.4.2.2 for details.
Opcode	2-3	0x0110	Command opcode.
Datalen	4-5	0x0	N/A
Return value/VFID	6-7		Some comments on specific errors (see Section 10.11.2.6 for the errors encoding): 0x0 = No Error. 0xD = EEXIST (no action, the device is already in non-PXE mode).
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16-31	0x0	Reserved.

10.11.2.7.8 Disable RX (0x000C)

This command is used to enable the driver to request a safe disable of Receive Enable.

Table 10-113. Disable RX Command

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section 10.11.2.4.1.1 for details.
Opcode	2-3	0x000C	Command opcode.
Datalen	4-5	0x0	No external buffer for this command.
Return value/VFID	6-7		Return value. Zeroed by driver. Written by firmware. See Table 10-120 .
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
Port	16		Indicates the port for which the operation is requested.
Reserved	17-31	0x0	Reserved.

The firmware should execute the following flow when receiving this command:

- Store OS2BMC enable bit.
- Clear OS2BMC enable bit.
- Clear RXEN
- Restore OS2BMC enable bit to original value.
- Return from command.

10.11.2.7.9 Get Firmware Event (Asynchronous Events) (0x0014)

This command is used by software to get the details of the last firmware events.

The E610 uses a firmware-triggered interrupt to signal software for asynchronous events. This interrupt is identified by bit 21 in the Extended Interrupt Cause Register (EICR). Firmware triggers the interrupt by writing to the Global Host Interrupt (GHINT) register.

Each PF can receive this interrupt independently. Before signaling the interrupt, firmware sets the PF Host Interface Control Register Event Pending bit (PF_HICR.EP) to 1b.

Upon receiving the interrupt or upon observing that PF_HICR.EP is set, software issues the Get Firmware Event command to determine which event has occurred. The typical events are the following:

- Link Status Event (0x0607)
- Health Status Event (0xFF22)
- Firmware Log Event (0xFF33)
- ASIC Temp TCA event (0x0C94)

Software repeatedly issues the Get Firmware Event command until all events have been read. Firmware clears the interrupt and PF_HICR.EP bit after software has exhausted the contents of the firmware buffer by repeated invocations of the Get Firmware Event command.

Table 10-114. Get Firmware Event Command

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section 10.11.2.4.1.1 for details.
Opcode	2-3	0x0014	Command opcode.
Datalen	4-5	0x0	Length in bytes of command buffer.
Return value/VFID	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16-31	0x0	Reserved.

Firmware responds to the Get Firmware Event command by writing the structure below to the PF_HIDA register.

Firmware copies the oldest event data from its internal buffer to the PF_HIDA_2 and PF_HIBA registers (if needed based on the event type and as would be indicated by bit0 in the "FW buffer status").

Table 10-115. Get Firmware Event Response

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section 10.11.2.4.1.2 for details.
Opcode	2-3	0x0014	Command opcode.
Datalen	4-5	0x0	Length in bytes of command buffer.
Return value/VFID	6-7		Return value; 0x0 = No error.
Cookie High	8-11	Cookie	Opaque value copied from the command message.
Cookie Low	12-15	Cookie	Opaque value copied from the command message.
FW Buffer Status	16-17		Bit 0: Set when an event was written to PF_HIDA_2; Clear otherwise. Bit 1: Set when there is additional event data pending in the firmware buffer; Clear otherwise. All other bits: Reserved.
Reserved	18-31	0x0	Reserved.

10.11.2.7.10 Program Fuse Bits (0xDD00)

This command is used during manufacturing to program a fuse.

After programming a fuse, firmware sets the corresponding *LVID* bit (except for the Configuration Signing fuse bits).

Fuse programming via this command is only allowed before the End of Manufacturing (EOM) condition has been met. Once EOM is satisfied, the only fuses that can be programmed are Key Revocation fuses and SREV fuses.

Table 10-116. Program Fuse Bits Command

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section 10.11.2.4.2.1 for details.
Opcode	2-3	0xDD00	Command opcode.
Datalen	4-5	0x40	Length in bytes of the command buffer.
Return value	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
Fuse ID	16-19		ID that uniquely identifies the fuse to program.
End of Manufacturing Indication	20:0	1 if EOBM should be set.	A flag to indicate that no more OEMIFP fuse programming is allowed. 0b = Do nothing. This flag must be 0 when the <i>Fuse ID</i> is non-zero. 1b = Set the <i>LVID</i> bit for all programmable OEMIFP fuses. When this parameter is non-zero, the <i>Fuse ID</i> must be 0.
Reserved	20.1-20.7	0x0	Reserved
Reserved	21-23	0x0	Reserved. Must be set to zero.
Data Address High	24-27	32 upper bits of the command buffer address	The FW will use Host DMA to copy the buffer to internal device memory.
Data Address Low	28-31	32 lower bits of the command buffer address	The FW will use Host DMA to copy the buffer to internal device memory.

Table 10-117. Program Fuse Bits Response

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section 10.11.2.4.2.2 for details.
Opcode	2-3	0xDD00	Command opcode.
Datalen	4-5	0x40	Length in bytes of the command buffer.
Return value	6-7		Return value; 0x0 = No error.
Cookie High	8-11	Cookie	Opaque value copied from the command message.
Cookie Low	12-15	Cookie	Opaque value copied from the command message.
Fuse ID	16-19		Programmed fuse ID. Copied from the command message.
Fuse Bit Width	20-23		Bit width of the value programmed. Ranges from 1 to 512.

Table 10-117. Program Fuse Bits Response [continued]

Name	Bytes.Bits	Value	Remarks
Reserved	24-31	0x0	Reserved. Must be set to zero.
Fuse Value	32-95		Programmed value. Read back after programming.

10.11.2.7.11 Read Fuse Bits (0xDD01)

This command is used during manufacturing to query a fuse's value.

Table 10-118. Read Fuse Bits Command

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section 10.11.2.4.2.1 for details.
Opcode	2-3	0xDD01	Command opcode.
Datalen	4-5	0x0	Length in bytes of the command buffer.
Return value	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
Fuse ID	16-19		ID that uniquely identifies the fuse to read.
Reserved	20-23		Reserved. Must be set to zero.
Data Address High	24-27	32 lower bits of the response buffer address.	1b = 1b = FW will use Host DMA to copy the response into this address.
Data Address Low	27-31	32 lower bits of the response buffer address.	FW will use Host DMA to copy the response into this address.

Table 10-119. Read Fuse Bits Response

Name	Bytes.Bits	Value	Remarks
Flags	1:0		See Section 10.11.2.4.2.2 for details.
Opcode	2-3	0xDD01	Command opcode.
Datalen	4-5	0x0	Length in bytes of the command buffer.
Return value	6-7		Return value; 0x0 = No error.
Cookie High	8-11	Cookie	Opaque value copied from the command message.
Cookie Low	12-15	Cookie	Opaque value copied from the command message.
Fuse ID	16-19		Read fuse ID. Copied from the command message.
End of Board Manufacturing Indication	20.0		The FW will return the EOBI status in this bit in the response.
LVID Set	20.1	0x0	The FW will return the LVID (lock bit) value for the requested fuse in the response.
Reserved	20.2-21	0x0	Ignored by FW.
Size of Fuse Field in Bits	22-23	0x0	The FW shall return the actual number of bits in the fuse field in the response. This can be used to query the size of a fuse field. In case of this usage, the buffer length should be set to at least 64-bytes.

Table 10-119. Read Fuse Bits Response [continued]

Name	Bytes.Bits	Value	Remarks
Data Address High	24-27		
Data Address Low	28-31		

10.11.3 Legacy Manageability Host Interface

This section details the legacy host interaction with the manageability portion of the E610. The information within this section is only available to the host driver, the BMC does not have access.

10.11.3.1 HOST CSR Interface (Function 1/0)

The software device driver of all functions communicates with the manageability block through CSR access. The manageability is mapped to address space 0x15800 to 0x15FFF on the target bus of each function.

Note: Writing to address 0x15800 from any function is targeted to the same address in the RAM.

10.11.3.2 Host Target Command Interface to Manageability

This interface is used by the software device driver for several of the commands and for delivering various types of data in both directions (Manageability-to-Host and Host-to-Manageability).

The address space is separated into two areas:

- Direct access to the internal data RAM: The internal shared (between firmware and software) RAM is mapped to address space 0x15800 to 0x15EFF. Writing/reading to this address space goes directly to the RAM.
- Control register is located at address 0x15F00

10.11.3.2.1 Host Target Command Interface Low-Level Flow

This interface is used for the external host software to access the manageability subsystem. Host software writes a command block or read data structure directly from the data RAM. Host software controls these transactions through a target access to the *HICR* control register (see [Section 8.2.2.22.4](#)).

The following flow shows the process of initiating a command to the manageability block:

1. The software clears the *FWSTS.FWRI* flag (clear by write one) to clear any previous firmware reset indications.
2. The software device driver takes ownership of the Management Host interface using the flow described in [Section 4.7](#).
3. The software device driver reads the HOST Interface Control register (see [Section 8.2.2.22.4](#)) and checks that the *Enable* (*HICR.EN*) bit is set.
4. The software device driver writes the relevant command block into the RAM area that is mapped to addresses 0x15800-0x15EFF.

5. The software device driver sets the *Command* (HICR.C) bit in the HOST Interface Control register (see [Section 8.2.2.22.4](#)). Setting this bit causes an interrupt to the ARC.
6. The software checks the FWSTS.FWRI flag to make sure a firmware reset didn't occur during the command processing. If this bit is set, the command may have failed.
7. The software device driver polls the HOST Interface Control register for the *Command* (HICR.C) bit to be cleared by firmware. The command should complete within half a second.
8. When firmware finishes with the command, it clears the *Command* (HICR.C) bit (if firmware replies with data, it should clear the bit only after the data is placed in the shared RAM area where the software device driver can read it).

If the software device driver reads the HOST Interface Control register and the HICR.SV bit is set to 1b, there is a valid status of the last command in the shared RAM. If the HICR.SV bit is not set, the command has failed with no status in the RAM.

On completion of access to the shared RAM software device driver should release ownership of the shared RAM using the flow described in [Section 4.7](#).

10.11.3.2.2 Host Interface Structure

10.11.3.2.2.1 Host Interface Command Structure

[Table 10-120](#) describes the structure used by the software device driver to send a command to firmware using the Host target command interface (shared RAM mapped to addresses 0x15800-0x15EFF).

Table 10-120. Host Driver Command Structure

Byte(s)	Description	Bit(s)	Value	Description
0	Command	7:0	Command Dependent	Specifies which host command to process.
1	Buffer Length	7:0	Command Length	Command Data Buffer length: 0 to 252, not including 32 bits of header.
2	Reserved	7:0		Reserved.
3	Checksum	7:0	Defined Below	Checksum signature. If the value is 0xFF, the checksum is not checked by the firmware.
255:4	Data Buffer	7:0	Command Dependent	Command Specific Data Minimum buffer size = 0 Maximum buffer size = 252

10.11.3.2.2.2 Host Interface Status Structure

[Table 10-121](#) lists the structure used by firmware to return a status to the software device driver via the Host target command interface. A status is returned after a command has been executed.

Table 10-121. Status Structure Returned to Host Driver

Byte(s)	Description	Bit(s)	Value	Description
0	Command	7:0	Command Dependent	Command ID.
1	Buffer Length	7:0	Status Dependent	Status buffer length: 252:0

Table 10-121. Status Structure Returned to Host Driver [continued]

Byte(s)	Description	Bit(s)	Value	Description
2	Return Status	7:0	Depends on Command Executing Results	0x1 = Status OK 0x2 = Illegal command ID 0x3 = Unsupported command 0x4 = Illegal payload length 0x5 = Checksum failed 0x6 = Data Error 0x7 = Invalid parameter 0x8-0x7F = Reserved 0x80-0xFF = Command Specific Errors. May be used by individual commands for command specific errors
3	Checksum	7:0	Defined Below	Checksum signature.
255:4	Data Buffer		Status Dependent	Status configuration parameters Minimum Buffer Size = 0 Maximal Buffer Size = 252 If Return Status is not "Status OK" the Data Buffer is empty.

10.11.3.2.2.3 Checksum Calculation Algorithm

The Host Command/Status structure is summed with this field cleared to 0b. The calculation is done using 8-bit unsigned math with no carry. The inverse of this sum is stored in this field (0b minus the result). Result: The current sum of this buffer (8-bit unsigned math) is 0b.

10.11.3.3 Host Interface Commands

10.11.3.3.1 Driver Info (0xDD)

This command is used to provide the driver information in NC-SI mode.

Table 10-122. Driver Info Host Command

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0xDD	Driver info command.
1	Buffer Length	7:0	0x5 + driver string length	Port Number + 4 bytes of the Driver info + driver string length (total length can be up to 255 bytes).
2	Reserved	7:0	0x0	Reserved.
3	Checksum	7:0		Checksum signature of the Host command.
4	Function Number	7:0	Function Number	Indicates the function currently reporting its driver info.
8:5	Driver Version	7:0	Driver Version	Numerical for driver version - should be: Byte 8 = Major Byte 7 = Minor Byte 6 = Build Byte 5 = SubBuild Note: The old version is kept to support CEM version 1 commands.
9:9 + driver string length	Driver String		Driver String	Driver string (not null terminated) as reported by driver. For example, in Linux the DRV_VERSION #define, or in NDIS, the OID_GEN_VENDOR_DRIVER_VERSION. Can be up to 250 bytes.

Following is the status returned on this command:

Table 10-123. Driver Info Host Status

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0xDD	Driver Info command.
1	Buffer Length	7:0	0x0	No data in return status.
2	Return Status	7:0	0x1	See Table 10-121 .
3	Checksum	7:0		Checksum signature.

10.11.3.3.2 Disable RXEN (0xDE)

This command is used to enable the driver to request a safe disable of Receive Enable.

Table 10-124. Disable RXEN Command

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0xDE	Driver info command.
1	Buffer Length	7:0	0x1	1 data byte attached to this command (the port number).
2	Reserved	7:0	0x0	Reserved.
3	Checksum	7:0		Checksum signature of the Host command.
4	Port Number	7:0	Port Number	Indicates the port for which the operation is requested.

Following is the status returned on this command:

Table 10-125. OS2BMC Control Status

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0xDE	Driver Info command.
1	Buffer Length	7:0	0x0	No data in return status.
2	Return Status	7:0	0x1	See Table 10-121 .
3	Checksum	7:0		Checksum signature.

The firmware should execute the following flow when receiving this command:

- Store OS2BMC enable bit.
- Clear OS2BMC enable bit.
- Clear RXEN
- Restore OS2BMC enable bit to original value.
- Return from command.

10.11.3.3.2.1 Shadow RAM Read (0x31)

This command is used to request a read from the Shadow RAM. Requesting addresses above Shadow RAM boundaries causes an error.

Table 10-126. Shadow RAM Read Command

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0x31	Shadow RAM Read.
2:1	Buffer Length	15:0	0x6	
3	Checksum	7:0		Checksum signature.
7:4	Address	31:0		Address to read from the Shadow RAM.
9:8	Length to Read	15:0		How many bytes to read. Can be up to 1024.

Table 10-127. Shadow RAM Read Response

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0x31	Shadow RAM Read.
1	Buffer Length (LS Byte)	7:0		Buffer length[7:0].
2	Buffer Length (MS bits)	7:5		Buffer Length[10:8].
2	Return Status	4:0		See Table 10-121 .
3	Checksum	7:0		Checksum signature.
7:4	Address	31:0		Address to read from the Shadow RAM.
9:8	Length Read	15:0		How many bytes were actually read.
11:10	Reserved			Reserved.
12: Length Read+12	Read Data			The Requested Data read from Shadow RAM.

10.11.3.3.2.2 Shadow RAM Write (0x33)

This command is used to update the Shadow RAM. It allows write to the writable parts of the Shadow RAM. If not all the area to write is writable, or not all the area is in the Shadow RAM range, the command returns an error.

Table 10-128. Shadow RAM Write Command

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0x33	Shadow RAM write.
2:1	Buffer Length	15:0	0x8 + Length to Write	
3	Checksum	7:0		Checksum signature.
7:4	Address	31:0		Address to write to the flash.
9:8	Length to Write	15:0		How many bytes to write. Can be up to 1024.
11:10	Reserved			
12: Length to Write+12	Write Data			The data to write to the Shadow RAM

Table 10-129. Shadow RAM Write Response

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0x33	Shadow RAM Write.
1	Buffer Length	7:0	0x6	
2	Return Status	7:0		See Table 10-121 .
3	Checksum	7:0		Checksum signature.
7:4	Address	31:0		Address to write to the Shadow RAM.
9:8	Data Written	15:0		How many bytes were actually written.

10.11.3.3.2.3 Shadow RAM Dump (0x36)

This command is used to trigger a Shadow RAM dump. It should be used after updating a module in Shadow RAM.

Table 10-130. Shadow RAM Dump Command

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0x36	Shadow RAM dump.
2:1	Buffer Length	15:0	0x0	
3	Checksum	7:0		Checksum signature.

Table 10-131. Shadow RAM Dump Response

Byte(s)	Name	Bit(s)	Value	Description
0	Command	7:0	0x36	Shadow RAM dump.
1	Buffer Length	7:0	0x0	
2	Return Status	7:0		See Table 10-121 .
3	Checksum	7:0		Checksum signature.

10.11.3.4 Software and Firmware Synchronization

Software and firmware synchronize accesses to shared resources in the E610 through a semaphore mechanism and a shared configuration register between the software device driver of the two ports and the firmware. This semaphore enables synchronized accesses to the following shared resources:

- NVM
- PHY 0 and PHY 1 registers
- MAC (LAN controller) shared registers. This semaphore protects the MAC registers described in [Section 8.2.2.15](#).
- I²C

The SW_FW_SYNC.REGSMP bit is used as a semaphore mechanism between software and firmware. Once software or firmware takes control over this semaphore flag, it can access the SW_FW_SYNC Register and claim ownership over specific resources. The SW_FW_SYNC includes pairs of bits (one owned by software and the other by firmware), where each pair of bits controls a different resource. A resource is owned by software or firmware when its respective bit is set. It is illegal to have both pair

bits set at the same time. Described below are the required sequences for gaining and releasing control over shared resources.

10.11.3.4.1 Gaining Control of Shared Resource by Software

1. The software device driver checks that the software device driver of the other LAN function does not use the software/firmware semaphore.
 - The software device driver polls the `SWSM.SMBI` bit till it is read as 0b, or time expires (recommended expiration is ~10 ms + expiration time used for the `SW_FW_SYNC.REGSMP`).
 - If the `SWSM.SMBI` is found at 0b, the semaphore is taken. Note that following this read cycle the hardware auto sets the bit to 1b.
 - If time expired, it is assumed that the software device driver of the other function malfunctioned. The software proceeds to the next step.
2. The software device driver checks that the firmware does not use the software/firmware semaphore and then takes its control.
 - Software polls the `SW_FW_SYNC.REGSMP` bit till it is read as 0b, or time expires (recommended expiration is ~50 ms). If time has expired the software assumes that the firmware malfunctioned and proceeds to the next step, while ignoring the firmware bits in the `SW_FW_SYNC` register.
3. The software takes control of the requested resource(s).
 - The software device driver reads the firmware and software bit(s) of the requested resource(s) in the `SW_FW_SYNC` register. If the bit(s) is cleared, the resource(s) is accessible (i.e. no other entity owns the resource(s)). In this case the software device driver sets the software bit(s) of the requested resource(s) in the `SW_FW_SYNC` register. The software then clears the `SW_FW_SYNC.REGSMP` and `SWSM.SMBI` bits (releasing the software/firmware semaphore register), and can use the specific resource(s).
 - Otherwise (i.e. either firmware or software of the other LAN function owns the resource), the software clears the `SW_FW_SYNC.REGSMP` and `SWSM.SMBI` bits and then repeats the entire process after some delay (recommended 5-10 ms).
 - If the resources are not released by the software device driver of the other LAN function in a timely manner (recommended expiration time is ~1 sec), the software device driver can assume that the other software device driver malfunctioned. In that case the software device driver should clear all software flags that it does not own (including `SW_FW_SYNC.REGSMP` bit) and then repeat the entire process once again.
 - If the resource is not released by the firmware (recommended expiration time for firmware is ~50 ms) the software device driver can assume that the firmware malfunctioned. In that case the software device driver should set the software bit(s) of the requested resource(s), while ignoring the corresponding firmware bits in the `SW_FW_SYNC` register.

Note: The firmware initializes its semaphore flags as part of its init flow. The software semaphores are not reset.

10.11.3.4.2 Releasing a Shared Resource by Software

1. The software device driver takes control over the software/firmware semaphore as described above for gaining shared resources.
2. The software device driver clears the bit(s) of the released resource(s) in the `SW_FW_SYNC` register.

3. The software device driver releases the software/firmware semaphore by clearing the SW_FW_SYNC.REGSMP and SWSM.SMBI bits
4. The software device driver should delay (recommended 5-10 ms) before trying to gain the semaphore immediately following its release.

10.11.3.4.3 Gaining Control of Shared Resource by Firmware

1. The firmware takes control over the software/firmware semaphore (SW_FW_SYNC register)
 - The firmware polls the SW_FW_SYNC.REGSMP bit till it is read as 0b, or timeout expires (recommended expiration time is ~10 ms).
 - If timeout has expired the firmware clears the SW_FW_SYNC.REGSMP bit (it is assumed the software device driver is not functional).
2. The firmware takes ownership of the requested resources
 - The firmware reads the software bit(s) corresponding to the requested firmware resource(s) in the SW_FW_SYNC register.
 - If the software bit is cleared (i.e. the software device driver does not own the resource), the firmware sets the firmware bit(s) of the requested resource(s). The firmware then clears the SW_FW_SYNC.REGSMP bit (releasing the software/firmware semaphore) and can use the specific resource(s).
 - Otherwise (i.e. software owns the resource), the firmware clears the SW_FW_SYNC.REGSMP bit and then repeats the above process after some delay (recommended delay of 5-10 ms).
 - If the resources owned by software are not released in a timely manner (~1 sec), the firmware “forces” its ownership over the requested resources. The firmware clears the software flags of the requested resources in the SW_FW_SYNC register (assuming the software that set those flags is not functional).

10.11.3.4.4 Releasing a Shared Resource by the Firmware

1. The firmware takes control over the software/firmware semaphore as described above for gaining shared resources.
2. The firmware clears the bit(s) of the selected resource(s) in the SW_FW_SYNC register.
3. The firmware releases the software/firmware semaphore by clearing the SW_FW_SYNC.REGSMP bit.
4. The firmware should delay before trying to gain the selected resource semaphore immediately following its release (recommended 5-10 ms).

10.12 Host Isolate Support

If a MC decides that malicious software prevents its usage of the LAN, it might decide to isolate the NIC from its driver. This is done using the TCO Reset command ([Section 10.6.4.12](#)).

If TCO isolate is enabled in the NVM, The TCO Isolate command disables PCIe write operations to the LAN port. As the driver needs to access the CSR space to provide descriptors to the NIC, this operation also stops the network traffic including OS-to-BMC and BMC-to-OS traffic as soon as the existing transmit and receive descriptor queues are exhausted.

Chapter 11 Electrical/Mechanical Specification

11.1 Introduction

This section describes E610 electrical and mechanical characteristics, including:

- Operating conditions
- Power Delivery
- Power Dissipation
- DC/AC specifications
- Package
- Supported devices

11.2 Operating Conditions

11.2.1 Absolute Maximum Ratings

Table 11-1. Absolute Maximum Ratings

Symbol	Parameter	Min	Max	Units
T_j	Junction Temperature Under Bias	-0	105	°C
T_{storage}	Storage Temperature Range	-40	105	°C

Notes:

- Stresses above those listed in the table can cause permanent device damage. These values should not be used as limits for normal device operation. Exposure to absolute maximum rating conditions for an extended period of time can affect device reliability.
- Maximum T_j of 125 °C allowed up to 4% of the time without affecting device reliability.

11.2.2 Recommended Operating Conditions

Table 11-2. Recommended Operating Conditions

Symbol	Parameter	Min	Typ	Max	Units
T_j	Junction Temperature			105	°C

Notes:

- For normal device operation, adhere to the limits in this table. Sustained operation of a device at conditions exceeding these values, even if they are within the absolute maximum rating limits, can result in permanent device damage or impaired device reliability.
- Device functionality is not guaranteed if conditions exceed recommended operating conditions.
- External Heat Sink (EHS) is needed for most applications.
- Thermal design power (TDP) is specified at 105 °C junction temperature.

11.3 Power Delivery

11.3.1 Power Supply Specification

Table 11-3. 3.3 V (VDD_3V3_IO) Power Supply Specification

VDD_3V3_IO (3.3 V) Parameters				
Parameter	Description	Min	Max	Units
Rise Time	Time from 10% to 90% mark.	1	10	ms
Monotonicity	Voltage dip allowed in ramp.	N/A	0	mV
Slope	Ramp rate at any given time between 10% and 90%. Min = $0.8 \cdot V(\text{min}) / \text{rise time (max)}$ Max = $0.8 \cdot V(\text{max}) / \text{rise time (min)}$	24	28,800	V/S
Operational Range	Voltage range for normal operating conditions (+-10%)	2.97	3.63	V
Ripple	Maximum voltage ripple (peak-to-peak).	N/A	VDD_3V3_IO ±200	mV
Overshoot	Maximum overshoot allowed.	N/A	100	mV
Overshoot Settling Time	Maximum overshoot allowed duration. Note: At that time delta voltage should be lower than 5 mV from steady state voltage.	N/A	0.05	ms

Notes:

- Used for IO and PHY LDO inputs.

Table 11-4. 1.8 V (VDD_1V8_IO) Power Supply Specification

VDDH Parameters				
Parameter	Description	Min	Max	Units
Rise Time	Time from 10% to 90% mark.	1	10	ms
Monotonicity	Voltage dip allowed in ramp.	N/A	0	mV
Slope	Ramp rate at any given time between 10% and 90%. Min = $0.8 \times V(\text{min}) / \text{rise time (max)}$ Max = $0.8 \times V(\text{max}) / \text{rise time (min)}$	0.33	7120	V/S
Operational Range	Voltage range for normal operating conditions (-3% +5%)	1.746	1.89	V
Ripple	Maximum voltage ripple (peak-to-peak).	N/A	54	mV
Overshoot	Maximum overshoot allowed.	N/A	0	mV

Notes:

- Used for PCIE, PHY, ETH PHY, PLLs, XTAL, GPIOs, PVT. POR cell and EFUSE programming.
- The overshoot is included in the max/min power value.

Table 11-5. 0.895 V (VDD_PLL, VDD_EPHY) Power Supply Specification

ETH_PHY (VP) and PLL (VDDREF/VDDPOST) Parameters				
Parameter	Description	Min	Max	Units
Rise Time	Time from 10% to 90% mark.	1	10	ms
Monotonicity	Voltage dip allowed in ramp.	N/A	0	mV
Slope	Ramp rate at any given time between 10% and 90%. Min = $0.8 \times V(\text{min}) / \text{rise time (max)}$ Max = $0.8 \times V(\text{max}) / \text{rise time (min)}$	0.33	7120	V/S
Operational Range	Voltage range for normal operating conditions (+-10%)	0.855	0.935	V
Ripple	Maximum voltage ripple (peak-to-peak). 600 KHz to 20 MHz.	N/A	45	mV
Overshoot	Maximum overshoot allowed.	N/A	50	mV
Overshoot Duration	Maximum overshoot allowed duration. Note: At that time delta voltage should be lower than 5 mV from steady state voltage.	0	0.05	ms

Table 11-6. 0.75 V (VDD) Power Supply Specification

Parameter	Description	Min	Max	Units
Rise Time	Time from 10% to 90% mark.	1	10	ms
Monotonicity	Voltage dip allowed in ramp.	N/A	0	mV
Slope	Ramp rate at any given time between 10% and 90%. Min = $0.8 \times V(\text{min}) / \text{rise time (max)}$ Max = $0.8 \times V(\text{max}) / \text{rise time (min)}$	0.33	7120	V/S
Operational Range	Voltage range for normal operating conditions (-10%/+5%)	0.695 ¹	0.7875	-
Ripple	Maximum voltage ripple (peak-to-peak).	N/A	37.5	mV

Table 11-6. 0.75 V (VDD) Power Supply Specification [continued]

Parameter	Description	Min	Max	Units
Overshoot	Maximum overshoot allowed.	N/A	50	mV
Overshoot Duration	Maximum overshoot allowed duration. Note: At that time delta voltage should be lower than 5 mV from steady state voltage.	0	0.05	ms

1. -7% for PCIe PHY.

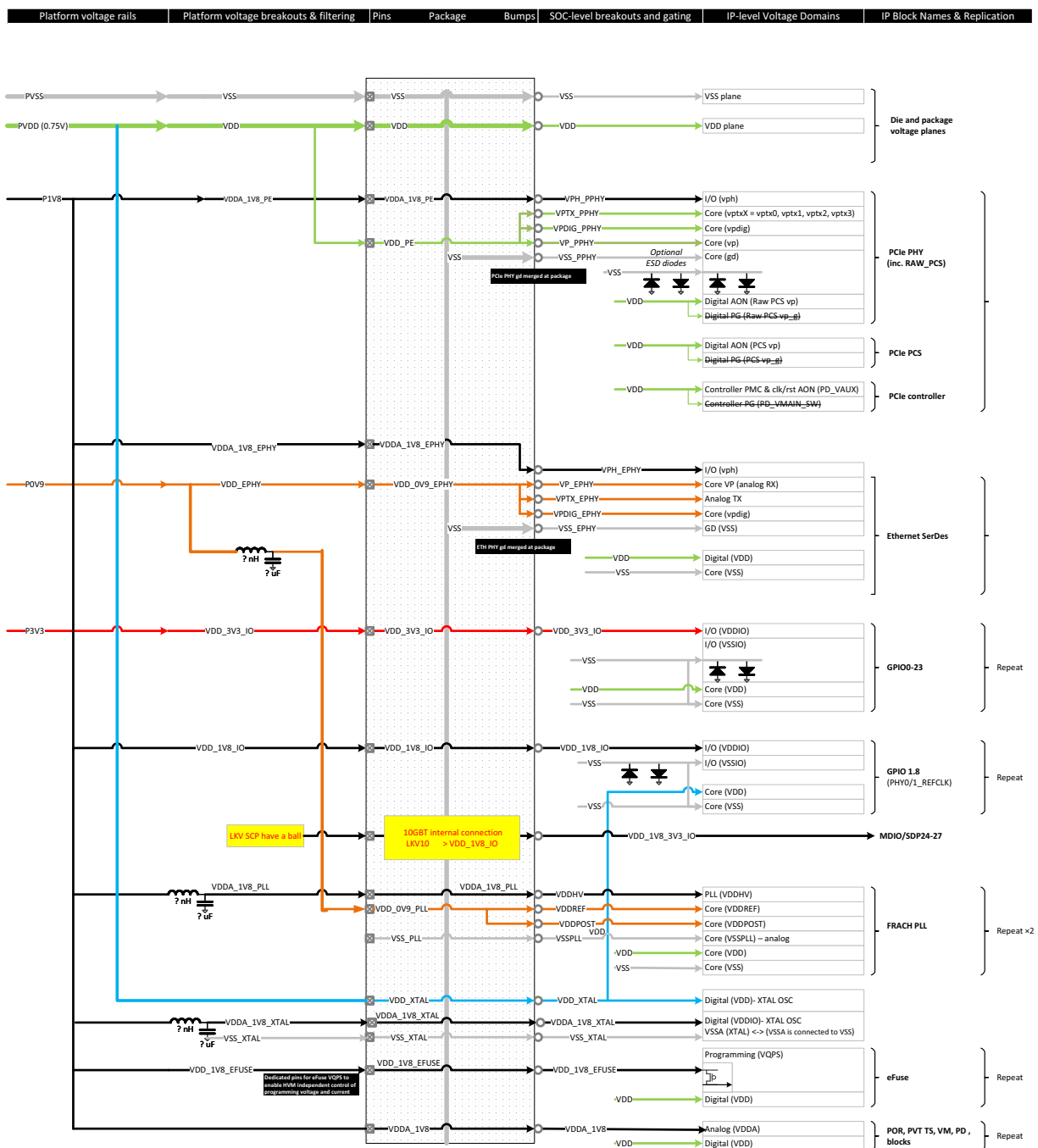
Table 11-7. 1000BASE-T Power Supply Specification

Symbol	Parameter	Condition	Min	Typical	Max	Units
V _{DDA} ¹	AVDD supply	For AVDD	0.825	0.85	0.876	V
V _{DDAC} ¹	AVDDC supply	For AVDDC	0.825	0.85	0.876	V
V _{DDAT} ²	AVDDAT supply	For AVDD33	2.231	2.3	2.369	V
V _{DDAS} ¹	AVDDS supply	For AVDDS	0.825	0.85	0.876	V
V _{DD} ¹	DVDD supply	For DVDD	0.631	0.65	0.67	V
V _{DDO} ¹	VDDO supply	For VDDO	1.71	1.8	1.89	V
IREF	Internal bias reference	Resistor connected to V _{ss}		4990 ± 1% Tolerance		Ω
T _A	Ambient operating temperature	Commercial Grade	0		70 ³	°C
		Industrial Grade	-40		+85	°C
T _J	Maximum junction temperature				105	°C

1. Maximum noise allowed on supplies is 25 mV peak-to-peak.

2. Maximum noise allowed on AVVDT is 10 mV peak-to-peak.

3. Commercial operating temperatures are typically below 70 °C (for example, 45 °C - 55 °C). The 70 °C max is specification limit.



11.3.1.1 Power On/Off Sequence

The following relationships between the application of the different power supplies should be maintained to avoid risk of either latch-up or forward-biased internal diodes:

- Application of power should begin with the 3.3 V (VDD_3V3_IO) and 1.8 V supplies. After 3.3 V reaches 80% of its final value, the digital core 0.75 V and 0.9 V supply as well as the 10GBASET PHY power supplies should be enabled and reach its final value within 10 ms.
- The E610 provides an internal power on reset (POR). An alternate external POR (LAN_POWER_GOOD) can be used to extend the reset if necessary.
- E610 initialization budgets 35 ms from system power-on to LAN_POWER_GOOD (or internal POR) completion. Remaining 65msec (for the PCIe spec 100msec system power to PERST# inactive) is allocated to internal initialization.
- The internal power on reset function monitors 1.8 V and 0.75 V only. Internal POR signal is held active after these supplies reach ~50%.
- Once all the power supplies are up, the pin straps are sampled to determine the platform configuration of the E610. Upon LAN_POWER_GOOD completion, the 25 MHz REFCLK oscillator is enabled. Depending on pin straps, the XTAL IP achieves locked status in ~4 ms (for crystal attached mode) or ~100 ns (for crystal bypass mode).
- 25 MHz REFCLK should be active to ensure that internal clock is active prior to internal POR complete. If this cannot be met, then LAN_POWER_GOOD signal should be used to extend reset. The 35 ms budget must still be met for PCIe timing.

Figure 11-2 illustrates a power supply turn-on order with timing that meets the system power on to PERST# inactive timing in the PCIe CEM specification. Of the 100 ms budget, 65 ms is reserved for internal initialization activity; Power supply delays and rise times are approximate. the key requirement is for power on reset (whether from the internal POR or LAN_POWER_GOOD) to be complete in 35 ms from system power stable.

For power down, it is recommended to turn off all rails at the same time and allow voltage to decay.

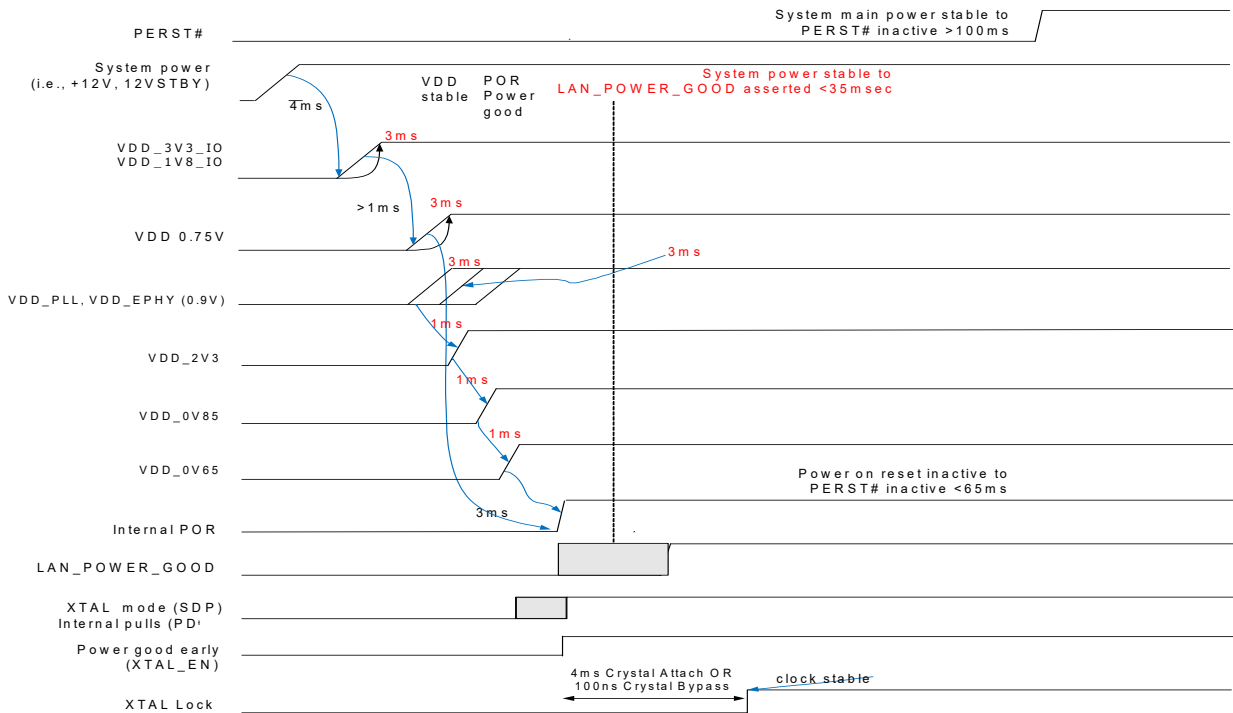


Figure 11-2. Power Supply Sequence

11.4 Power Dissipation

The following tables list the targets for device power consumption. The numbers listed apply to device current and power and do not include power losses on external components.

Power numbers are provided in two modes:

- **MAX-Power (TDP):** Power consumption of the device specified at maximum recommended operating junction temperature on fast silicon process corner, and nominal power supply voltage. This power should be used for thermal and power supply design.
- **Typical-Power:** Power consumption of the device at nominal operation conditions: typical silicon, nominal power supply voltage, and $T_j = 80^\circ\text{C}$.

11.4.1 MAX Power (TDP)

Table 11-8. MAX Power -

	Link Speed	
	2x10G PCIe Gen4x4	2x10G PCIe Gen3x1
VDD_3V3_IO (A)	0.033	0.033
VDD_1V8_IO (A)	0.381	0.244
VDD 0.75V (A)	0.787	0.396
VDD_PLL, VDD_EPHY (0.9V) (A)	0.174	0.170
AVDDT 2V3 PHY (A)	0.427	0.325
AVDD 0V85 PHY (A)	0.864	0.658
DVDD 0V65 PHY (A)	1.21	0.342
Power (W)	4.04 ¹	2.53

1. TDP power were measured on Typical units

11.4.2 Typical Power

Table 11-9. Typical Active Power

	Link Speed	
	2x10G PCIe Gen4x4	
VDD_3V3_IO (A)	0.042	
VDD_1V8_IO (A)	0.366	
VDD 0.75V (A)	0.72	
VDD_PLL, VDD_EPHY (0.9V) (A)	0.173	
AVDDT 2V3 PHY (A)	0.34	
AVDD 0V85 PHY (A)	0.82	
DVDD 0V65 PHY (A)	1.24	
Power (W)	3.78	

Note: Typical conditions: typical material, T_J = 80 °C, nominal voltages. and continuous network traffic at link speed.

Table 11-10. Typical Idle Power

	Link Speed	
	2x10G PCIe Gen4x4	
VDD_3V3_IO (A)	0.042	
VDD_1V8_IO (A)	0.366	
VDD 0.75V (A)	0.64	
VDD_PLL, VDD_EPHY (0.9V) (A)	0.172	
AVDDT 2V3 PHY (A)	0.34	
AVDD 0V85 PHY (A)	0.82	
DVDD 0V65 PHY (A)	1.24	
Power (W)	3.7	

Note: Typical conditions: typical material, T_J = 80 °C, nominal voltages. and no traffic.

11.5 DC/AC Specifications

11.5.1 Digital I/O DC Specifications

Table 11-11. Digital Functional 3.3V I/O DC Electrical Characteristics

Symbol	Parameter	Conditions	Min	Max	Units
VOH	Output High Voltage		2.4		V
VOL	Output Low Voltage			0.4	V
IOH	Output High Current		12		mA
IOL	Output Low Current		12		mA
VIH	Input High Voltage		2.0	VDD_3V3_IO +300 mV	V
VIL	Input Low Voltage		-0.3	0.8	V
Iil	Input Current ¹	VI=3.3 V / 0 V		10	μA
PU	Internal pull-up		10	20	KΩ

1. Input Leakage Current

11.5.1.1 Open Drain I/O DC Specification

This section applies to SMBD, SMBCLK, SMBALRT_N, PE_WAKE_N.

Table 11-12. Open Drain I/O DC Characteristics

Symbol	Parameter	Condition	Min	Max	Units
Vih	Input High Voltage		2.0	VDD_3V3_IO +0.3	V
Vil	Input Low Voltage		-0.2	0.8	V
Ileakage	Output Leakage Current ¹	$0 \leq V_{in} \leq VDD\_3V3\_IO$ max		100	μA
Vol	Output Low Voltage	@ Ipullup = 4 mA	0.4		V
Cin	Input Pin Capacitance				pF
Ioffsmb	Input leakage Current ¹	VDD_3V3_IO off or floating		100	μA
IOL	Output Current Low		6		mA

1. Device meets this specification whether powered or un-powered.

11.5.1.2 NC-SI I/O DC Specification

Table 11-13. NC-SI I/O DC Characteristics

Symbol	Parameter	Conditions	Min	Typical	Max	Units
Vref ¹	Bus High Reference		3.0	3.3	3.6	V
Vabs	Signal Voltage Range		-0.3		3.765	V
Vil	Input Low Voltage				0.8	V
Vih	Input High Voltage		2.0			V
Vol	Output Low Voltage	I _{ol} = 4 mA, Vref = Vref _{min}	0		0.4	V
Voh	Output High Voltage	I _{ol} = -4 mA, Vref = Vref _{min}	2.4		Vref	V
Iih	Input High Current	Vin = 3.6 V, Vref = 3.6 V	0		200	μA
Iil	Input Low Current	Vin = 0 V Vref = Vref _{min} to Vref _{max}	-20		0	μA
Ioh	Input High Current		12			mA
Iol	Input Low Current		12			mA
Vckm	Clock Midpoint Reference Level				1.4	V
Pin Capacitance	Cin		1.5	1.7		pF
Iz	Leakage Current for Output Signals in High-Impedance State	0 ≤ Vin ≤ Vih _{max} Vref = Vref _{max}	-20		20	μA

1. Vref = Bus high reference level. This parameter replaces the term “supply voltage” since actual devices might have internal mechanisms that determine the operating reference for the sideband interface that are different from the devices overall power supply inputs. Vref is a reference point that is used for measuring parameters such as overshoot and undershoot and for determining limits on signal levels that are generated by a device. In order to facilitate system implementations, a device must provide a mechanism (e.g. a power supply pin, internal programmable reference, or reference level pin) to allow Vref to be set to within 20 mV of any point in the specified Vref range. This is to enable a system integrator to establish an interoperable Vref level for devices on the sideband interface. Although the NC-SI spec define the Vref_{max} up to 3.6 V, the E610 supports the Vref_{max} up to 3.46 V (3.3 V +5%).

11.5.2 Digital I/F AC Specifications

11.5.2.1 Digital I/O AC Specifications

Table 11-14. Digital 3.3 V I/O AC Electrical Characteristics

Parameters	Description	Min	Max	Condition
F _{max}	Maximum Operating Frequency		100 MHz, 300 MHz	100 MHz at 25 pF load. 300 MHz at 8 pF load.
T _{or}	Output Rise Time	0.5 ns	6 ns	
T _{of}	Output Fall Time	0.5 ns	6 ns	

11.5.2.2 SMBus and I²C AC Specifications

The E610 meets the SMBus AC specification as defined in the *System Management Bus (SMBus) Specification*, Version 2, Section 3.1.1 (<http://www.smbus.org/specs/index.html>) and the I²C specification.

The E610 also supports a 400 KHz SMBus and I²C (as a secondary), and meets the specifications listed in Table 11-15:

Table 11-15. Support for 400 KHz SMBus

Symbol	Parameter	Min	Typical	Max	Units
F_{SMB}	SMBus Frequency	10		400	KHz
T_{BUF}	Time Between Stop and Start	1.44			μs
$T_{HD,STA}$	Hold Time After Start Condition. After This Period, the First Clock is Generated.	0.48			μs
$T_{SU,STA}$	Start Condition Setup Time	1.6			μs
$T_{SU,STO}$	Stop Condition Setup Time	1.76			μs
$T_{HD,DAT}$	Data in Hold Time	0.32			μs
$T_{SU,DAT}$	Data in Setup Time	0.1			μs
T_{LOW}	SMBClk Low Time	0.8			μs
T_{HIGH}	SMBClk High Time	1.44			μs

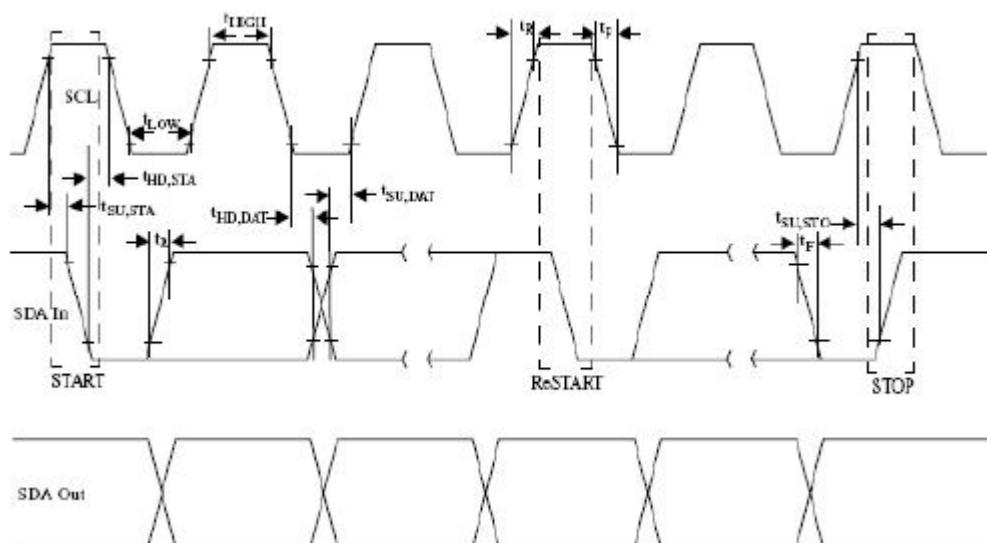


Figure 11-3. SMBus I/F Timing Diagram

11.5.2.3 FLASH AC Specification

The E610 is designed to support a serial Flash. Applicable over recommended operating range with $C_{load} = 16$ pF (unless otherwise noted). For Flash I/F timing specifications, see [Table 11-16](#) and [Figure 11-4](#).

Note: [Table 11-16](#) applies to FLSH_SI, FLSH_SO, FLSH_SCK and FLSH_CE_N.

Table 11-16. Flash I/F Timing Parameters

Symbol	Parameter	Min	Typical	Max	Units
t_{SCK}	FLSH_SCK Clock Frequency ¹	0		50	MHz
t_{RI}	FLSH_SO Rise Time		3		ns
t_{FI}	FLSH_SO Fall Time		3		ns
t_{WH}	FLSH_SCK High Time ²		8		ns
t_{WL}	FLSH_SCK Low Time ²		12		ns
t_{CS}	FLSH_CE_N High Time	10			ns
t_{CSS}	FLSH_CE_N Setup Time	5			ns
t_{CSH}	FLSH_CE_N Hold Time	5			ns
t_{SU}	Data-in Setup Time	2			ns
t_H	Data-in Hold Time	3			ns
t_V	Output Valid			7	ns
t_{HO}	Output Hold Time	2			ns
t_{DIS}	Output Disable Time			7	ns

1. Nominal 40%:60% high:low duty cycle.

2. Measured at VDD_3V3_IO/2.

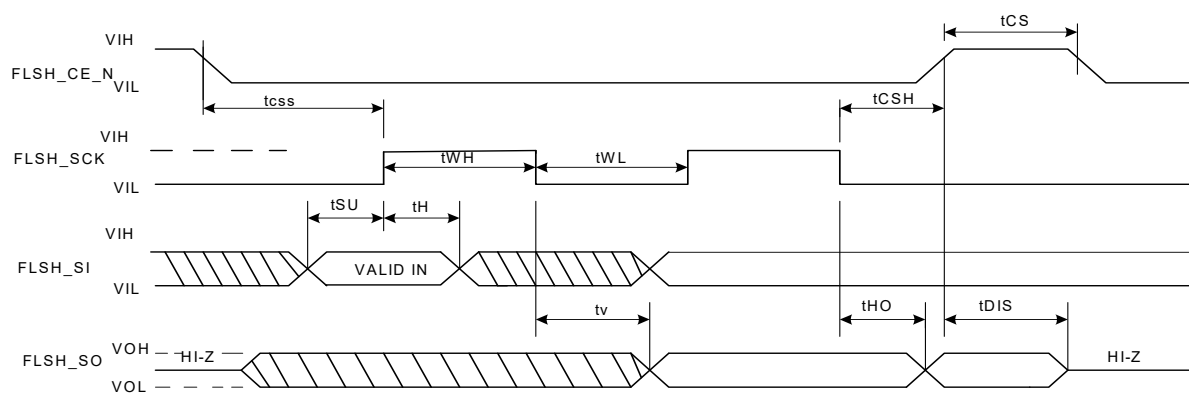


Figure 11-4. Flash I/F Timing Diagram

11.5.2.4 NC-SI AC Specifications

The E610 supports the NC-SI standard as defined in the DMTF *Network Controller Sideband Interface (NC-SI) Specification*. The NC-SI timing specifications can be found in [Table 11-17](#) and [Figure 11-5](#).

Table 11-17. NC-SI Interface AC Specifications

Symbol	Parameter	Min.	Typical	Max	Units
	REF_CLK Frequency		50	50+100 ppm	MHz
	REF_CLK Duty Cycle ¹	35		65	%
T_{co}	Clock-to-Out ^{2,3} ($10\text{ pF} \leq C_{load} \leq 50\text{ pF}$)	2.5		12.5	ns
T_{skew}	Skew Between Clocks			1.5	ns
T_{su}	TXD[1:0], TX_EN, RXD[1:0], CRS_DV, RX_ER Data Setup to REF_CLK Rising Edge ³	3			ns
T_{hd}	TXD[1:0], TX_EN, RXD[1:0], CRS_DV, RX_ER Data Hold From REF_CLK Rising Edge ³	1			ns
T_r/T_f	Signal Rise/Fall Time ⁴	1		6	ns
T_{ckr}/T_{ckf}	REF_CLK Rise/Fall Time ⁵	0.5		3.5	ns
T_{pwrz}	Interface Power-Up High Impedance Interval	2			μ s
T_{pwrt}	Power Up Transient Interval (recommendation)			100	ns
V_{pwrt}	Power Up Transient Level (recommendation)	-200		200	mV
T_{pwre}	Interface Power-Up Output Enable Interval			10	ms
$T_{clkstrt}$	EXT_CLK Startup Interval			100	ms

1. REF_CLK duty cycle measurements are made from Vckm to Vckm. Clock skew T_{skew} is measured from Vckm to Vckm of two NC-SI devices and represents maximum clock skew between any two devices in the system.
2. This timing relates to the output pins timing while T_{su} and T_{hd} relate to timing at the input pins.
3. All timing measurements are made between Vckm and Vm. All output timing parameters are measured with a capacitive load between 10 pF and 50 pF.
4. Rise and fall time are measured between points that cross 10% and 90% of Vref (see [Table 11-13](#)). The middle points (50% of Vref) are marked as Vckm and Vm for clock and data, respectively.
5. A serial 33 Ω resistor might be required in case of very short trace on the board.

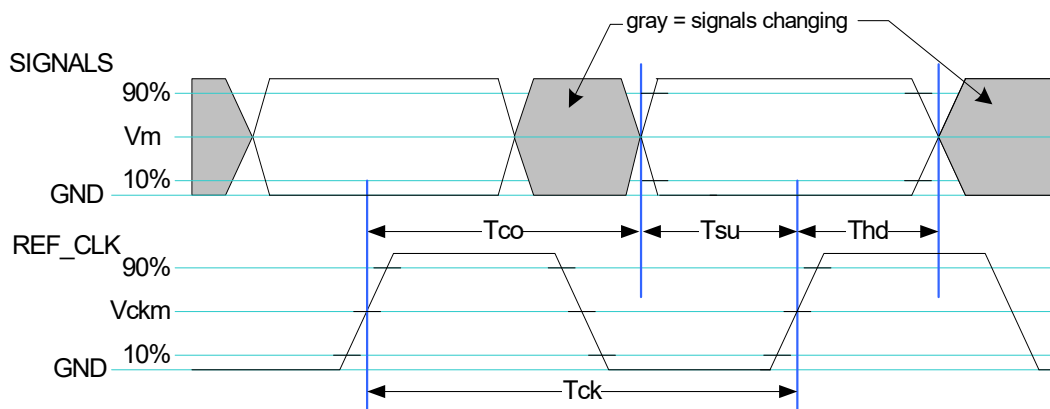


Figure 11-5. NC-SI AC Timing Diagram

11.5.2.5 JTAG AC Specification

The E610 is designed to support the IEEE 1149.1 standard. The following timing specifications are applicable over recommended operating range. For JTAG I/F timing specifications, see Table 11-18 and Figure 11-6.

Table 11-18 applies to JTCK, JTMS, JTDI and JTDO.

Table 11-18. JTAG I/F Timing Parameters¹

Symbol	Parameter	Min	Typical	Max	Units
t _{JH}	JTMS and JTDI Hold Time	10			ns
t _{JSU}	JTMS and JTDI Setup Time	10			ns
t _{JPR}	JTDO Propagation Delay			15	ns

1. Timing measured relative to JTCK reference voltage of VDD_3V3_IO/2.

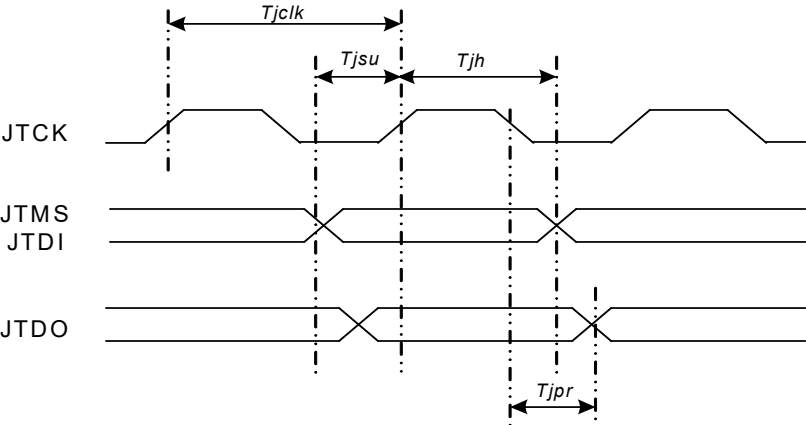


Figure 11-6. JTAG AC Timing Diagram

11.5.2.6 MDIO AC Specification

The E610 is designed to support the MDIO specifications defined in IEEE 802.3 clauses 22 and 45. The following timing specifications are applicable over recommended operating range with $C_{load} = 16$ pF (unless otherwise noted). For MDIO I/F timing specifications, see [Table 11-19](#), [Figure 11-7](#), and [Figure 11-8](#).

[Table 11-19](#) applies to MDIO0, MDC0, MDIO1 and MDC1.

Table 11-19. MDIO I/F Timing Parameters¹

Symbol	Parameter	Min	Typical	Max	Units
t_{MCLK}	MDC Clock Frequency	2.4		24	MHz
t_{MH}	MDIO Hold Time	10			ns
t_{MSU}	MDIO Setup Time	10			ns
t_{MPR}	MDIO Propagation Delay	10		30	ns

1. Timing measured relative to MDC reference voltage of 2.0 V (V_{ih}).

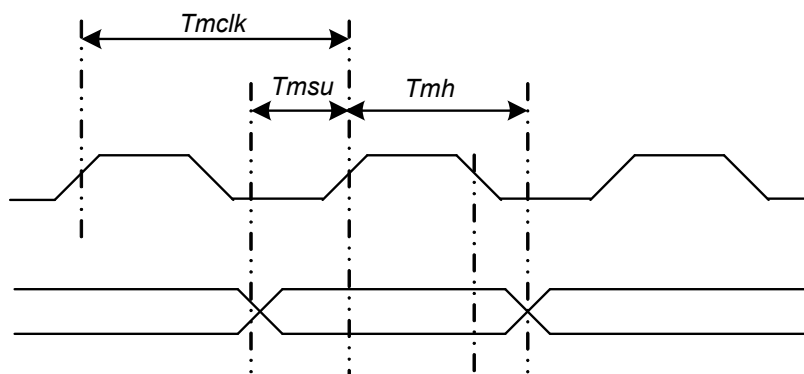


Figure 11-7. MDIO Input AC Timing Diagram

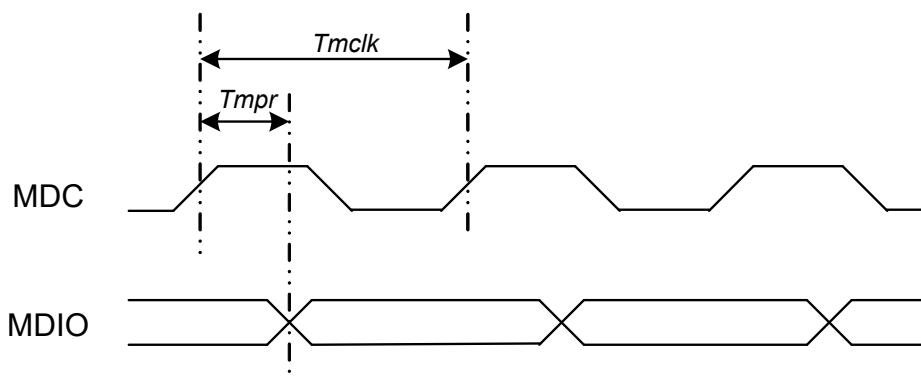


Figure 11-8. MDIO Output AC Timing Diagram

11.5.2.7 Reset Signals

For power-on indication, the E610 can either use an internal power-on circuit that monitors the VDD and VDDH power supplies, or an external reset using the LAN_PWR_GOOD pin. The POR_BYPASS pin defines the reset source. The timing between the power up sequence and the different reset signals is described in [Section 11.3.1.1](#).

Note: A schema of the power on logic can be found in [Figure 11-9](#).

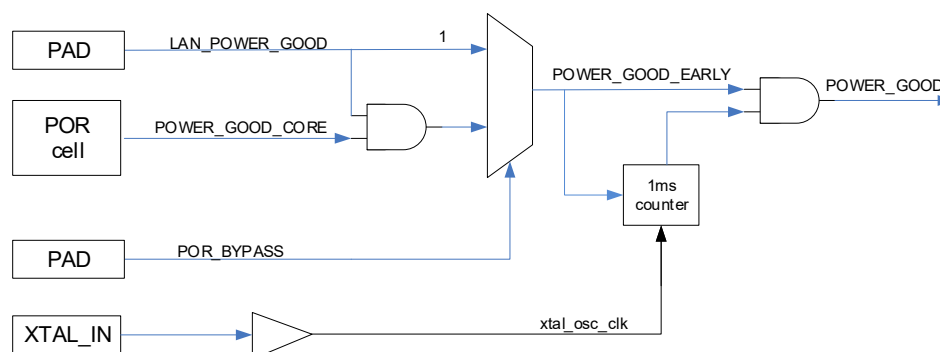


Figure 11-9. Schema of Power-On Logic

11.5.2.7.1 Power-On Reset BYPASS

When asserting the POR_BYPASS pad, the E610 uses the LAN_PWR_GOOD pin as power-on indication. Otherwise the E610 uses a logical OR of the internal power on detection circuit and the LAN_PWR_GOOD pin in order to generate the internal power on reset signal.

11.5.3 PCIe Interface AC/DC Specification

The E610 PCIe interface supports the PCIe Gen 5.0 electrical specification defined in:

- *PCIe Express Base Specification*, Revision 5.0
- *PCI Express Card Electromechanical Specification*, Revision 5.0

11.5.4 Network Interface AC/DC Specification

The E610 Ethernet interface supports a variety of operating modes, as defined in [Section TBD](#).

11.5.5 Reference Clock Specification

The E610 requires reference clocks for E610 Core clocking, and a differential reference clock for the Ethernet PHY and the PCIe PHY. In addition, the 1000BASE-T and 10GBASE-T components require dedicated reference clocking support. The electrical specifications for these reference clocks is documented in the sub-sections that follow.

11.5.5.1 XTAL Reference Clock Specification

The E610 Core main reference clock can be provided either with a 25 MHz oscillator or 25 MHz crystal attached to the E610.

Table 11-20. Crystal Input Reference Clock Electrical Characteristics

Description	Symbol	Min	Typical	Max	Units
Frequency Range (crystal mode)		18		40	MHz
Frequency Range (bypass mode)		0		100	MHz
Cycle-to-Cycle Jitter	CCJ	-100		100	ps
Rise Time	t_R		40	80	ps
Fall Time	t_F		40	80	ps
Output Duty Cycle	t_{DO}	40	50	60	%
Lock Time	t_{LOCK}			100,000	cycles
Input Voltage Threshold (bypass mode)					% VDDA
Area (100 μm [X] by 85.02 μm [Y])	A		0.0085		sq. mm
Total Power (unloaded) ¹	P		1	1.4	mW
Output Load	C_L			50	fF
Operating Junction Temperature	T_j	-40	25	125	°C
Digital Power Supply ²	VDD	0.675	0.75	0.825	V
Analog Power Supply	VDDA	1.62	1.8	1.98	V

1. The power is achieved at Frequency = 40 MHz, XGAIN1 = 0, XGAIN0 = 0, Crystal with ESR = 30 Ω , Shunt capacitance = 2 pF, and Load capacitance = 4 pF.
2. This macro supports functional operation with digital power supply down to 0.55 V.

11.5.5.2 Ethernet Reference Clock Specification

The Ethernet PHY REFCLK input buffer supports HCSL voltage levels and contains internal AC coupling capacitors and 50 ohm termination. Best performance may be achieved by using an HCSL drive oscillator with direct connections to the RECLK inputs (no other component connected on the circuit board).

Table 11-21. Reference Clock Amplitude Requirements

Symbol	Parameter	Min	Max	Units	Conditions
VDREF_CLK	Differential input swing for ref_pad_clk_{p,m}	300		mVpp	Differential inputs must meet VDREF_CLK, VDSEREF_CLK_DC, and VDSEREF_CLK_AC.
VDREF_CLK_ALT	Differential input swing for ref_alt_clk_{p,m}	200		mVpp	Differential inputs must meet VDREF_CLK_ALT, VDSEREF_CLK_DC, and VDSEREF_CLK_AC.
VDSEREF_CLK_DC	Input voltage level	-0.15	0.85	V	Differential inputs must meet both VDREF_CLK and VDSEREF_CLK_DC or both VDREF_CLK_ALT and VDSEREF_CLK_DC.
VDSEREF_CLK_AC ¹	Absolute voltage level at pin including overshoot	-0.3	1.15	V	Differential inputs must meet both VDREF_CLK and VDSEREF_CLK_AC or both VDREF_CLK_ALT and VDSEREF_CLK_AC.

1. Allowed overshoot time = 10% of the reference clock period.

Table 11-22. Reference Clock Input Requirements

Symbol	Parameter	Min	Max	Units	Conditions
FREF_CLK ¹	Reference clock frequency	19.2	200	MHz	Not continuous as per protocol.
FREF_OFFSET	Reference clock frequency offset	-100	100	ppm	
RMSJREF_CLK <= 8 Gbps	Reference clock random jitter		1.6	ps _{ms}	Integrated RJ from 12 KHz to 20 MHz,
			1.2		Integrated Rj from 2 KHz to 20 MHz.
DJREF_CLK <= 8 Gbps	Reference clock deterministic jitter		2.8	ps, pp	0.75-10 MHz (offset) ²
			5.6		0.2-50 MHz (offset) ²
RMSJREF_CLK <= 16 Gbps, > 8 Gbps	Reference clock random jitter		1.0	ps _{ms}	Integrated RJ from 12 KHz to 20 MHz,
			0.8		Integrated Rj from 2 KHz to 20 MHz.
DJREF_CLK <= 16 Gbps, > 8 Gbps	Reference clock deterministic jitter		1.7	ps, pp	0.75-10 MHz (offset) ²
			3.4		0.2-50 MHz (offset) ²
RMSJREF_CLK > 16 Gbps	Reference clock random jitter		0.45	ps _{ms}	Integrated RJ from 12 KHz to 20 MHz,
			0.3		Integrated Rj from 2 KHz to 20 MHz.
DJREF_CLK > 16 Gbps	Reference clock deterministic jitter		0.4	ps, pp	0.75-10 MHz (offset) ²
			0.85		0.2-50 MHz (offset) ²
DCREF_CLK	Duty cycle	40	60	%	
VCMREF_CLK	Common mode input level	0	VP	V	Differential inputs.
VDREF_CLK	Differential input swing for ref_pad_clk_{p,m}	300		mVpp	Differential inputs must meet VDREF_CLK, VDSEREF_CLK_DC, and VDSEREF_CLK_AC.
VDREF_CLK_ALT	Differential input swing for ref_alt_clk_{p,m}	200		mVpp	Differential inputs must meet VDREF_CLK_ALT, VDSEREF_CLK_DC, and VDSEREF_CLK_AC.
VDSEREF_CLK_DC	Input voltage level	-0.15	0.85	V	Differential inputs must meet both VDREF_CLK and VDSEREF_CLK_DC or both VDREF_CLK_ALT and VDSEREF_CLK_DC.
VDSEREF_CLK_AC	Absolute voltage level at pin including overshoot	-0.3	1.15	V	Differential inputs must meet both VDREF_CLK and VDSEREF_CLK_AC or both VDREF_CLK_ALT and VDSEREF_CLK_AC.
SKEWDIF_CLK	Differential clock input skew		100	ps	
SWREF_CLK	Differential clock input skew rate	0.6		V/ns	20-80%

1. Allowed overshoot time - 10% of the FREF_CLK period.

2. DJ noise limits are for noise within an offset from the divided reference clock and harmonics of the divided reference clock.

11.5.5.3 PCIe Reference Clock Specification

The PCIe PHY REFCLK input buffer supports HCSL voltage levels and contains internal AC coupling capacitors and 50 Ω termination. Best performance might be achieved by using an HCSL drive oscillator with direct connections to the REFCLK inputs (no other component connected on the circuit board). It must follow the PCIe reference clock specifications as outlined in the following tables.

Table 11-23. PCIe Reference Clock Requirements

Application	Reference Clock Jitter and AC Requirements
PCIe 1.1 — 2.5 Gbps	Refer to the PCI Express 3.0 Card Electromagnetic Specification.
PCIe 2.1 — 5.0 Gbps	Refer to the PCI Express 3.1 base specification.
PCIe 3.1 — 8.0 Gbps	Refer to the PCI Express 3.1 base specification.
PCIe 4.0 — 16.0 Gbps	Refer to the PCI Express 4.0 base specification.

Table 11-24. Reference Clock Amplitude Requirements

Symbol	Parameter	Min	Max	Units	Conditions
VDREF_CLK	Differential input swing for ref_pad_clk_{p,m}	300		mVpp	Differential inputs must meet VDREF_CLK, VDSEREF_CLK_DC, and VDSEREF_CLK_AC.
VDSEREF_CLK_DC	Input voltage level for ref_pad_clk_{p,m}	-0.15	0.85	V	Differential inputs must meet both VDREF_CLK and VDSEREF_CLK_DC.
VDSEREF_CLK_AC ¹	Absolute voltage level at pin including overshoot for ref_pad_clk_{p,m}	-0.3	1.15	V	Differential inputs must meet both VDREF_CLK and VDSEREF_CLK_AC.
VDREF_CLK_ALT	Differential input swing for ref_alt_clk_{p,m}	200		mVpp	Differential inputs must meet VDREF_CLK_ALT, and VDSEREF_CLK_ALT.
VDSEREF_CLK_ALT	Input voltage level for ref_alt_clk_{p,m}	0	VP	V	Differential inputs must meet VDREF_CLK_ALT, and VDSEREF_CLK_ALT.

1. Allowed overshoot time = 10% of the reference clock period.

11.5.5.4 Reference Clock Implementation

There are two types of reference clock implementations:

- External off-chip reference clocks using `ref_pad_clk_p` and `ref_pad_clk_m`.
- On-chip reference clocks using `ref_alt0_clk_p/m` and `ref_alt1_clk_p/m`.

Both reference clock implementations support either a differential clock or a CMOS single-ended clock. The reference clock input is an unterminated, AC coupled input.

If the reference clock is a single-ended CMOS clock, one of the clock inputs (such as `ref_pad_clk_p`) must be a full swing CMOS clock, and the other input (such as `ref_pad_clk_m`) should be tied to either high or low.

If the reference clock is differential, the Host Clock Signal Level (HCSL) is the preferred signaling specification. It can also be Common Mode Logic (CML), Low Voltage Differential Signaling (LVDS). If the signaling used is other than HCSL, special care must be taken to meet the common mode level specification.

An example showing level shift of LVDS signal to the common mode range supported by the REFCLK input is show in [Figure 11-10](#). The R1, R2, C1 should be chosen to achieve the desired common mode termination and bandwidth.

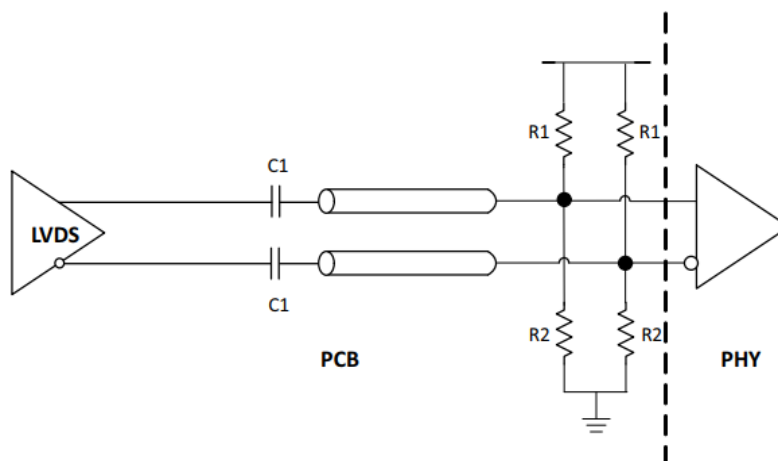


Figure 11-10. Example of Reference Clock LVDS Driver Implementation

Any reference clock inputs that are not being used should be tied to ground.

The selected reference clock(s), either on-chip or external, can be used in generating the MPLL clock for either direct clocking mode or cascaded clocking mode. For PCIe Gen1 and Gen2, the reference clock can also be used as the input reference for the lane ROPLL. In cascaded clocking mode, the ROPLL uses high-speed reference clock supplied from the MPLL output.

11.5.5.5 10GBASE-T REFCLK Requirements

Table 11-25. XTAL_IN / XTAL_OUT Timing

Parameter	Symbol	Conditions	Min	Typical	Max	Units
Reference clock frequency	Fclk			25		MHz
Reference clock frequency tolerance asynchronous			-50		+50	ppm
Reference clock input voltage swing differential pk-pk	A		400	750	900	mVppd
Reference clock duty cycle	Tduty		45	50	55	%
Reference clock rise and fall times	tr,tf	20% to 80% of amplitude	0.3	0.5	0.8	ns
Reference clock jitter	tj	Integrated from 1 to 30Mhz			0.5	ps, RMS
Differential input termination resistance (external)	Zin		80	100	110	Ω
Input Return Loss	SDD11	differential, 100 Ω			-12	$\delta\beta$

Note: Customers can select (through REFCLK_SEL pin) between two reference clock frequencies: 25 MHz and 156.25 MHz. The crystal oscillator option only supports 25 MHz

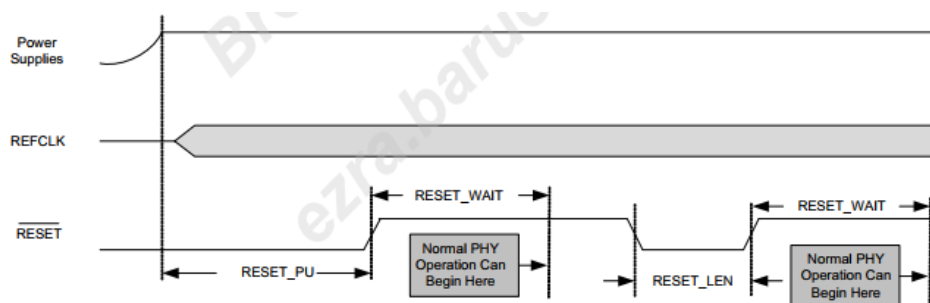


Figure 11-11. Recommended Operating Conditions

11.6 Package Characteristics

11.6.1 Mechanical Configuration

The E610 is assembled in a 13.75x12.25 mm FCCSP MCP package with 6-layer substrates. Refer to [Section 11.7](#) for the package mechanical drawings.

Table 11-26. Package Specifications

SKU	Body Size	Ball Count	Ball Pitch	Ball Matrix	Substrate
TBD	16 x 14 mm	412	Minimal 0.5 mm	Hybrid with mixed ball pitches	3-2-3

11.6.2 Heat Sink Mechanical Load Limits

This section presents guidelines for the maximum loads associated with heat sink to PCB attachment.

Loads are applied on the entire raised portion of the lid by a rigid heat sink or rigid actuator that is parallel to the lid plane. Short-term and long-term loads are applied slowly to prevent dynamic loads from being created. The maximum permissible BGA compression is 15% (TBD) of the assembled BGA height.

Table 11-27. Mechanical Load Limits

Load	Maximum
Short-term compressive load (applied for up to two minutes)	30 gm per solder ball. 76 psi on the raised portion of the lid.
Long-term compressive load	15 gm per solder ball. 39 psi on the raised portion of the lid.
Dynamic load	TBD 30g shock per MIL-STD-810F, Method 516.5 Procedure II.

11.6.3 Thermal Resistance Parameters

For complete E610 package thermals, refer to the Flowtherm model.

Table 11-28. Typical Thermal Resistance Parameters

Parameter	Value
Thermal resistance, junction to case, θ_{JC}	10.83 °C /W
Thermal resistance, junction to PCB, θ_{JB}	6.98 °C /W

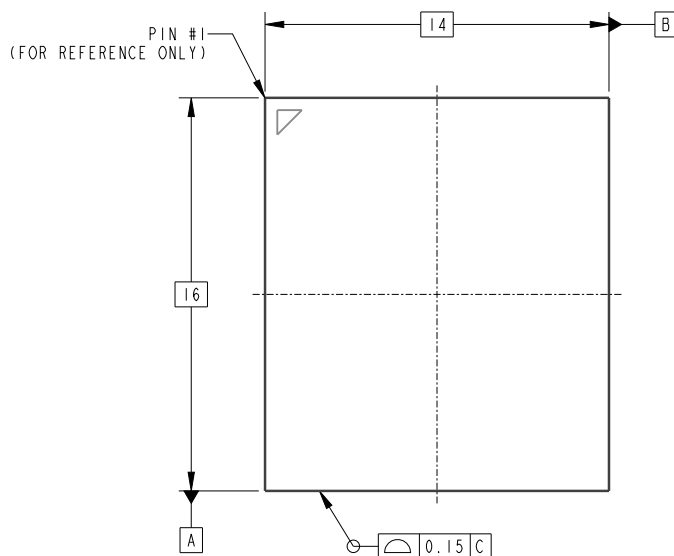
11.6.4 Electrical

Package electrical models are part of the IBIS files.

11.7 Package Mechanical Drawings

Notes:

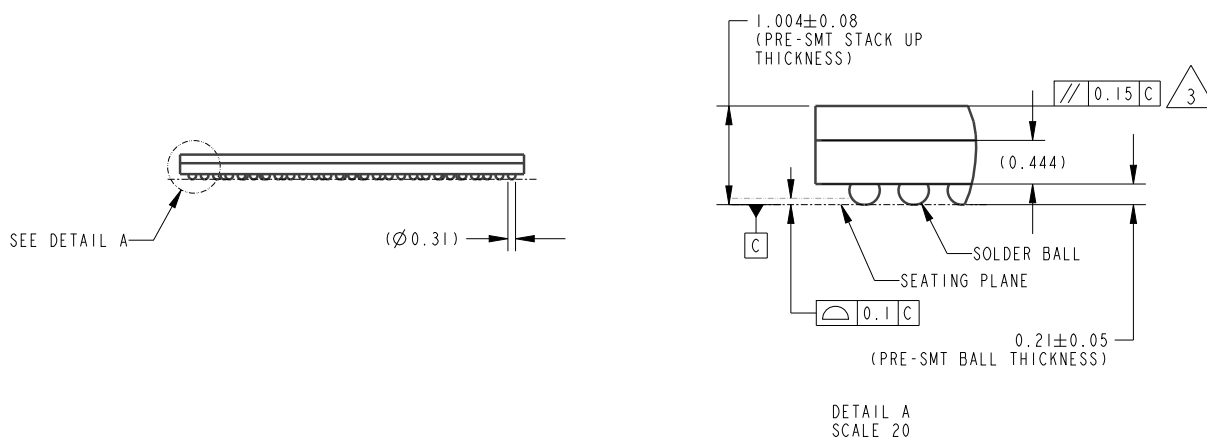
- Unless otherwise specified, interpret dimensions and tolerances in accordance with ASME Y14.5-2009. Dimensions are in millimeters.
- Unless otherwise specified, all tolerances are: Linear $\pm 0.1\text{mm}$, Angular $\pm 0.5^\circ$.



NOTES. UNLESS OTHERWISE SPECIFIED

1. ALL Z STACK UP HEIGHT ESTIMATES ARE BASED ON PRE SMT BALL HEIGHT.
2. PRIMARY DATUM C AND SEATING PLANE ARE DEFINED BY THE SPHERICAL CROWNS OF THE SOLDER BALLS.
3. PARALLELISM MEASUREMENT SHALL EXCLUDE ANY EFFECT OF MARK ON TOP SURFACE OF THE PACKAGE.
4. THIS DIMENSION IS MEASURED AT THE MAXIMUM DIAMETER OF THE TERMINAL (BALL) IN A PLANE PARALLEL TO PRIMARY DATUM C
5. BALL PLACEMENT USE 0.300mm SOLDER BALL.

Figure 11-12. Mechanical Package Diagram (Top View)



NOTES: UNLESS OTHERWISE SPECIFIED

1. ALL Z STACK UP HEIGHT ESTIMATES ARE BASED ON PRE SMT BALL HEIGHT.
2. PRIMARY DATUM C AND SEATING PLANE ARE DEFINED BY THE SPHERICAL CROWNS OF THE SOLDER BALLS.
3. PARALLELISM MEASUREMENT SHALL EXCLUDE ANY EFFECT OF MARK ON TOP SURFACE OF THE PACKAGE.
4. THIS DIMENSION IS MEASURED AT THE MAXIMUM DIAMETER OF THE TERMINAL (BALL) IN A PLANE PARALLEL TO PRIMARY DATUM C
5. BALL PLACEMENT USE 0.300mm SOLDER BALL.

Figure 11-13. Mechanical Package Diagram (Side View)



1. ALL Z STACK UP HEIGHT ESTIMATES ARE BASED ON PRE SMT BALL HEIGHT.
2. PRIMARY DATUM C AND SEATING PLANE ARE DEFINED BY THE SPHERICAL CROWNS OF THE SOLDER BALLS.
3. PARALLELISM MEASUREMENT SHALL EXCLUDE ANY EFFECT OF MARK ON TOP SURFACE OF THE PACKAGE.
4. THIS DIMENSION IS MEASURED AT THE MAXIMUM DIAMETER OF THE TERMINAL (BALL) IN A PLANE PARALLEL TO PRIMARY DATUM C
5. BALL PLACEMENT USE 0.300mm SOLDER BALL.

Figure 11-14. Mechanical Package Diagram (Bottom View)

11.8 Devices Supported

11.8.1 Flash

The E610 requires Flash devices with the following characteristics:

- Quad SPI interface
- 24-bits address size
- Nominal 3.3 V power supply and I/O voltage levels
- Max frequency equal or above 66 MHz
- Supports Serial Flash Discoverable Parameters (SFDP)
- Minimum density: 16 MB
- *Quad Enable* bit set in the status register by default, or set during the initial programming along with the Flash descriptor.

Table 11-29. Supported Flash Parts

Manufacturer	128 Mbit (16 MB)
TBD	TBD
TBD	TBD
TBD	TBD
TBD	TBD
TBD	TBD

NOTE: *This page intentionally left blank.*

Chapter 12 Design Considerations and Guidelines

This section provides guidelines for selecting components, connecting interfaces, using special pins, and layout guidance.

Unused interfaces should be terminated with pull-up or pull-down resistors. These are indicated in [Section 2](#) or reference schematics.

Some unused interfaces must be left open. Do not attach pull-up or pull-down resistors to any balls identified as No Connect or Reserved No Connect.

12.1 Connecting the PCIe Interface

The E610 connects to the host system using a PCIe interface. The interface can be configured to operate in several link modes as detailed in [Section 3](#). A link between the ports of two devices is a collection of lanes. Each lane must be AC-coupled between its corresponding transmitter and receiver, with the AC-coupling capacitor located close to the transmitter side (within 1 inch). Each end of the link is terminated on the die into nominal 100 differential DC impedance. Board termination is not required.

Refer to the *PCI Express* Base Specification, Revision 4.0*, and *PCI Express* Card Electromechanical Specification*.

12.1.1 Link Width Configuration

The E610 supports link widths of x4, x2, or x1 as determined by the Flash *Lane Width* field in the PCIe initialization configuration. The configuration is loaded using bits 9:4 in the *Max Link Width* field of the Link Capabilities register (0xAC); default setting is 0x04. The E610 default link width is x4.

During link configuration, the platform and the E610 negotiate a common link width. For proper operation, the selected maximum number of PCIe lanes must be connected to the host system.

12.1.2 Polarity Inversion and Lane Reversal

To ease routing, designers have the flexibility to the lane reversal modes supported by the E610. Polarity inversion can also be used, since the polarity of each differential pair is detected during the link training sequence.

When lane reversal is used, some of the down-shift options are not available. For a description of available combinations, see [Section 3](#).

12.1.3 PCIe Reference Clock

The E610 requires a 100 MHz differential reference clock called PE_CLK_p and PE_CLK_n. These signals are typically generated on the system board and routed to the PCIe connector. For add-in cards, the clock is furnished with the PCIe connector.

Note: The frequency tolerance for the PCIe reference clock is +/- 300 ppm.

12.1.4 Bias Resistor

For proper biasing of the PCIe analog interface, a 4.99 K Ω 1% resistor needs to be connected between P0_IREF /P1_IREF and ground. A 200 Ω 1% resistor needs to be connected between PE_RESREF/ETH_RESREF and ground. To avoid noise coupled onto this reference signal, place the bias resistor close to the E610 and keep traces as short as possible.

12.1.5 Miscellaneous PCIe Signals

The E610 signals power management events to the system by pulling low the PE_WAKE_N signal. This signal operates like the PCI PME# signal. Note that somewhere in the system, this signal has to be pulled high to the auxiliary 3.3 V supply rail.

The PE_RST# signal, which serves as the familiar reset function for the E610, needs to be connected to the host system's corresponding signal.

12.1.6 PCIe Layout Recommendations

For information regarding the PCIe signal routing, refer to the *Intel PCIe Design Guide*.

12.2 Connecting the 10GBASE-T MDI Interfaces

In 10GBASE-T mode, the line interface on the E610 can drive up to 100 meters of CAT-6a unshielded twisted pair or 100 meters of CAT-7 shield cable (100 Ω differential impedance). It can also drive 55 meters of CAT-6 cable. In 1000BASE-TX and 100MBASE-TX modes, it can drive 130 meters of CAT-5e (or better) cable. It is designed to drive this via a quad, 50 Ω center tapped 1:1 transformer connected to an RJ-45 PCB-mount jack. Solutions that combine the transformer and RJ-45 jack into a single device are also supported.

The line interface on the E610 supports automatic A/B and C/D pair swaps and inversions (MDI-X). It also supports provisioned ABCD to DCBA pair reversal for ease of routing via an NVM setting, which sets this configuration at power up.

Note: This reversal does not swap polarities thus A+ maps to D+, etc.

12.2.1 MDI Circuit Guidelines

The MDI discrete design and integrated magnetic components were chosen for inclusion in the reference design and Bill of Material (BOM). Refer to [Section 12.13](#) for more details. These components can deliver the performance required for this demanding application.

12.2.2 Magnetics Module

The magnetics module has a critical effect on overall IEEE and emissions conformance. The E610 should meet the performance required for a design with a reasonable margin to allow for manufacturing variation. Carefully qualifying new magnetics modules prevents problems that might arise because of interactions with other components or the Printed Circuit Board (PCB) itself. The magnetics specified should comply with the specifications listed in the Intel® 10-GBASE-T Magnetic Specification Electrical/Mechanical Requirements for 10GBASE-T Magnetic Components, which includes separate specifications for discrete and integrated magnetics modules.

The steps involved in magnetics module qualification are:

1. Verify that the vendor's published specifications in the component datasheet meet or exceed the required IEEE 802.3an specifications and the internal specifications listed in the Intel® 10-GBASE-T Magnetic Specification Electrical/Mechanical Requirements for 10GBASE-T Magnetic Components.
2. Independently measure the component's electrical parameters on a test bench, checking samples from multiple lots. Check that the measured behavior is consistent from sample-to-sample and that measurements meet the published specifications.
3. Perform physical layer conformance testing and EMC (FCC and EN) testing in real systems. Vary temperature and voltage while performing system level tests.

Magnetics modules for 10GBASE-T Ethernet as used by the E610 are similar to those designed for 1GBASE-T, except that the electrical requirements for the board layout and magnetics are more stringent. Refer to [Section 12](#) for specific electrical requirements that the magnetics need to meet.

12.2.3 MDI Layout Guidance

The MDI that was chosen for E610 designs consists of an RJ-45 right-angle PCB jack, Bob Smith termination and discrete magnetics (see [Figure 12 -1](#)). Magnetics can be implemented either as a discrete module or an integrated solution combining magnetics and the RJ-45 jack.

Minimizing the amount of space needed for the PHY is important because other interfaces compete for physical space on a Network Interface Card (NIC) or LAN on Motherboard (LOM) near the connector. The PHY circuits need to be as close as possible to the connector.

[Figure 12 -1](#) illustrates some basic placement distance guidelines. It shows four differential pairs, and the layout can be generalized for a 10 GbE system with four analog pairs. The ideal placement for the E610 is approximately two inches behind the magnetics module for both the discrete and integrated solutions.

12.2.3.1 Discrete Magnetics

The E610 uses voltage-drive PHY, therefore, there is no need for a separate voltage source that connects to the center tap of transformer. the impedance of the Bob Smith termination is 75 Ω Channel A, B, C and D, whereas the PHY input impedance is 100 Ω .

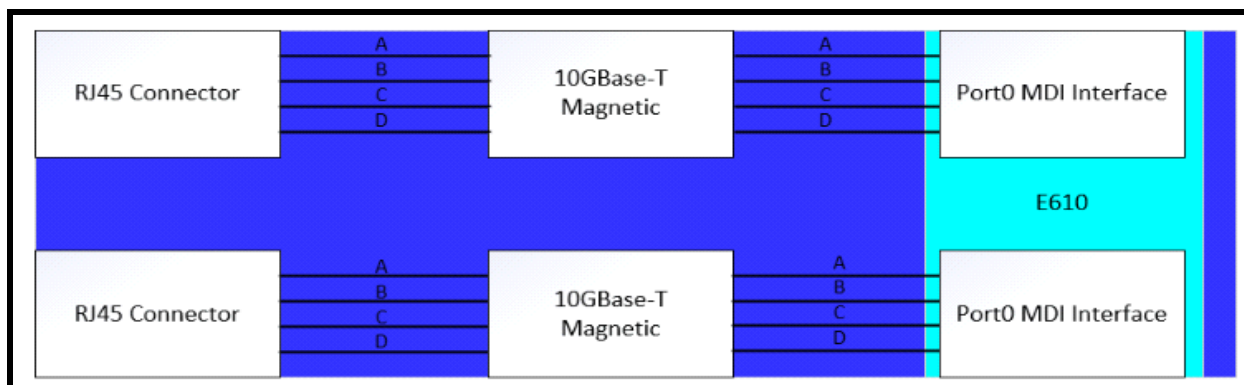


Figure 12-1. Basic MDI Placement Guidelines

The E610, referred to as a LAN silicon in [Figure 12-2](#) and [Figure 12-3](#), must be at least 1.5 inches from the I/O back panel. To help reduce EMI, the following recommendations should be followed:

- Minimize the length of the MDI interface.
- Place the MDI traces no closer than 0.5 inch (1.3 cm) from the board edge.

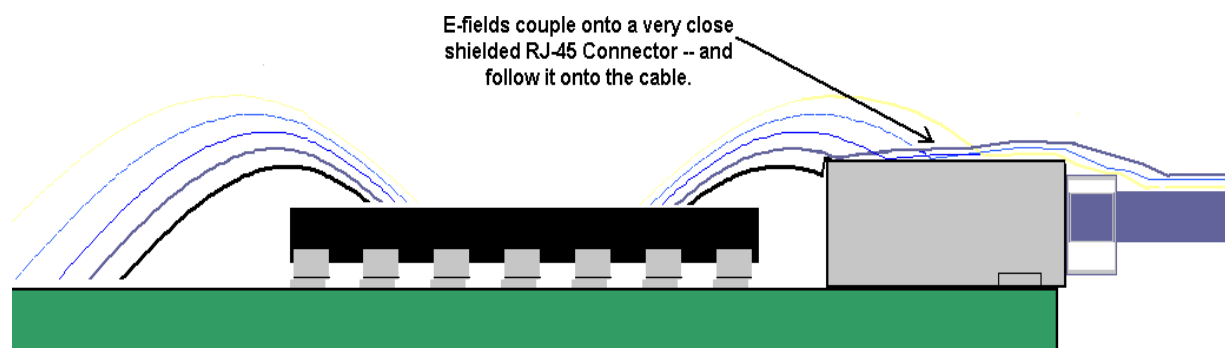


Figure 12-2. Effect of Device Placed Less Than One Inch from the RJ-45 Connector

The associated grounding scheme (blue) with the front-end of the chip is shown in the microstrip traces between the transformer and the RJ-45 (see [Figure 12-3](#)).

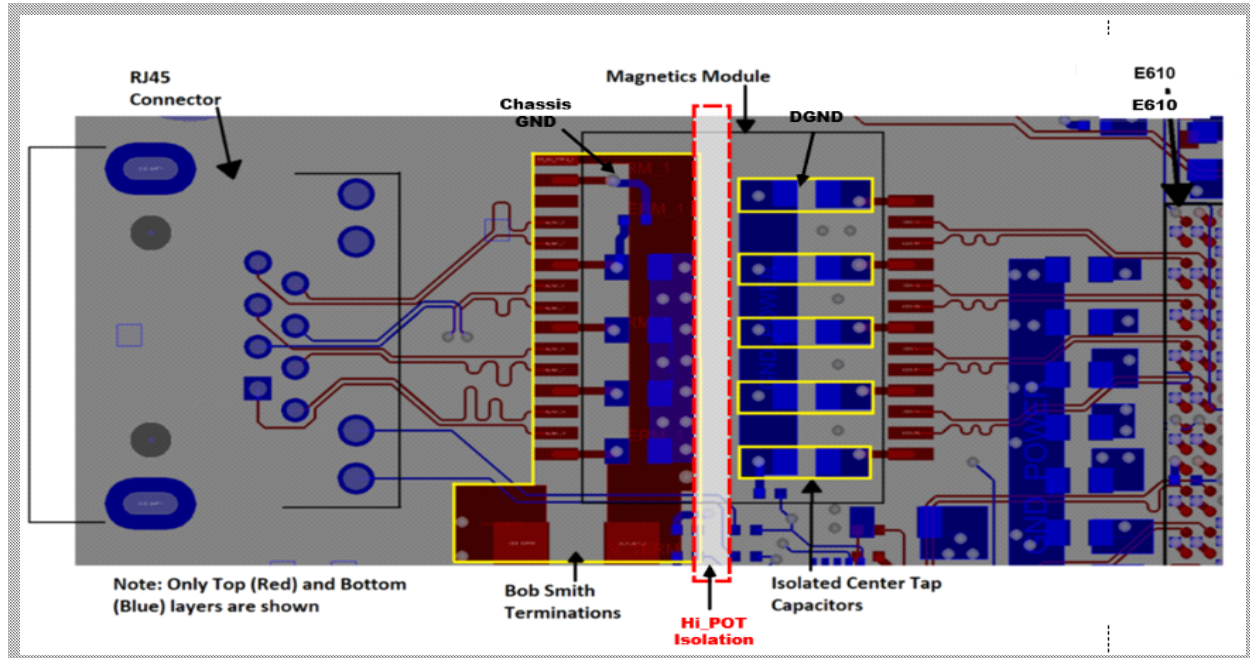


Figure 12-3. Transformer Bypass Components

From these figures (Figure 12 -3), the following should be noted:

- The ground is split underneath the magnetics, with the chassis ground being present under the front-end (Figure 12-3), and the circuit ground under the E610 side of the magnetics. Thus, the MDI traces on the line side are referenced to chassis ground.
- The magnetic module is used with the common mode choke facing the E610. This enables a Bob Smith termination to be implemented from the line side transformer center taps to chassis ground.
- Bob Smith termination and the shield of the RJ-45 jack are both connected to chassis ground, which are the two large circular metalizations in the MDI.
- The resistors in the Bob Smith termination are required to be 0805s to handle the cable discharge voltages.
- Similarly, the magnetics should be placed as close as possible to the RJ-45 jack.
- In Figure 12 -3, the Hi-POT clearance for cable discharge is shown and designed with ≥ 80 mils of clearance from the ground. Note that the breakdown voltage in FR-4 is lowest in the x-y axes planar to the PCB and highest in the z-axis between layers.

12.2.3.2 Integrated Magnetics

The guidance on MDI trace implementation between the E610 and the integrated module remains the same as with discrete magnetics, including length requirements and target impedance.

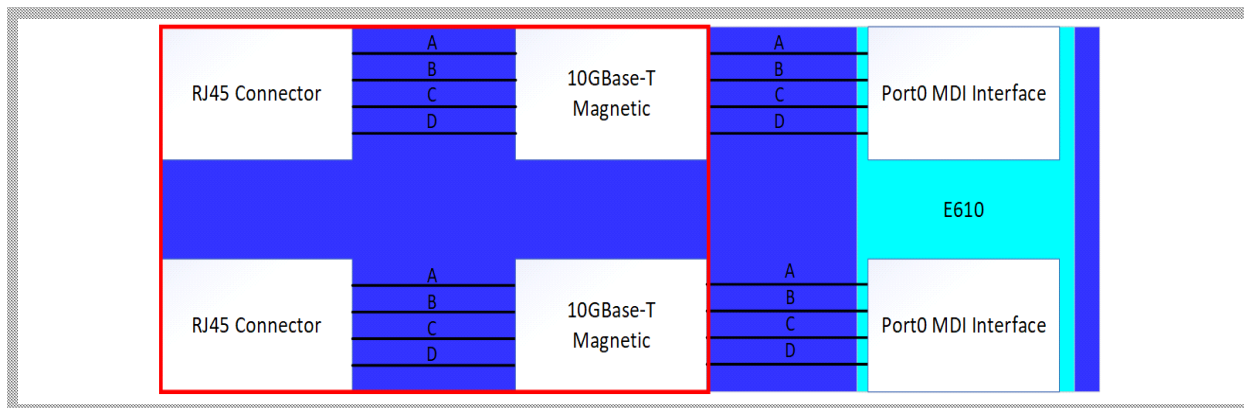


Figure 12-4. Basic MDI Placement Guidelines

Note: The RJ-45 jack and 10GBase-T Magnetics (red outline in Figure 12-4) are included in the integrated magnetics module.

The differences between the discrete and integrated magnetics implementations are as follows:

- The ground split that occurs underneath discrete magnetics modules is performed internally when using integrated magnetics. As a result, this ground split does not need to be addressed.
- The required Bob Smith termination is implemented internally when using integrated magnetics.

12.2.3.3 MDI Differential Pair Trace Routing for LAN Design

Trace routing considerations are important to minimize the effects of crosstalk and propagation delays on sections of the board where high-speed signals exist. Signal traces should be kept as short as possible to decrease interference from other signals, including those propagated through power and ground planes.

12.2.3.4 Signal Trace Geometry

One of the key factors in controlling trace EMI radiation are the trace length and the ratio of trace-width to trace-height above the reference plane. To minimize trace inductance, high-speed signals and signal layers that are close to a reference or power plane should be as short and wide as practical. Ideally, the trace-width to trace-height above the ground plane ratio is between 1:1 and 3:1. To maintain trace impedance, the width of the trace should be modified when changing from one board layer to another if the two layers are not equidistant from the neighboring planes.

Each pair of signals should target a differential impedance of 100 Ω . $\pm 15\%$.

A set of trace length calculation tools are made available from Intel to aid with MDI topology design. Contact your Intel representative for tool availability.

When designing a board layout, the automatic router feature of the CAD tool must not route the differential pairs without intervention. In most cases, the differential pairs require manual routing.

Note: Measuring trace impedance for layout designs targeting 100 Ω often results in lower actual impedance due to over-etching. Designers should verify actual trace impedance and adjust the layout accordingly. If the actual impedance is consistently low, a target of 105 Ω to 110 Ω should compensate for over-etching.

12.2.3.4.1 Matching Traces within a Pair (P and N)

P and N for each MDI pair should be matched to within 5 mils on the PCB to prevent common-to-differential and differential-to-common conversion due to the length mismatch.

If in-pair length matching is not possible using bends or small loops, serpentine routing (zigzag of a shorter trace) is acceptable if only one to three meanders are routed within 200 mils of the source of the skew or the end(s) of any otherwise unmatched lengths of a differential trace segment. For example, near device pins and/or at or near the connector pins and possibly at differential signal vias. Refer to the Intel® Ethernet Controller E610 Checklists for more details.

Table 12-1. MDI Routing Summary

Parameter	Main Route Guidelines	Breakout Guidelines ¹	Notes
Signal group	MDI_P[3:0] MDI_N[3:0]		
Microstrip*/Stripline* uncoupled single-ended impedance specification	50 Ω $\pm$ 10%		
Microstrip/Stripline uncoupled differential impedance specification	100 Ω $\pm$ 15%		^{2,3}
Microstrip nominal trace width	Design dependent	Design dependent	
Microstrip nominal trace space	Design dependent	Design dependent	³
Microstrip/Stripline trace length	< 8 inches		Table 12-2
Microstrip/Stripline pair-to-pair space (edge-to-edge)	$\geq$ 7 times the dielectric thickness		
Microstrip/Stripline bus-to-bus spacing	$\geq$ 7 times the dielectric thickness		
Matching traces within a pair (P and N)	< 5 mils		
Keep pair-to-pair length differences	< 2 inches		

1. Pair-to-pair spacing $\geq$ 7 times the dielectric thickness for a maximum distance of 500 mils from the pin. The phase tolerance between MDI_P and MDI_N is <5mils.
2. Board designers should ideally target 100 Ω $\pm$ 10%. If it's not feasible (due to board stackup) it is recommended that board designers use a 95 Ω $\pm$ 10% target differential impedance for MDI with the expectation that the center of the impedance is always targeted at 95 Ω . The $\pm$ 10% tolerance is provided to allow for board manufacturing process variations and not lower target impedances. The minimum value of impedance cannot be lower than 90 Ω .
3. Simulation shows 80 Ω differential trace impedances degrade MDI return loss measurements by approximately 1 dB from that of 90 Ω .

Table 12-2. Maximum Trace Lengths Based on Trace Geometry and Board Stackup

Dielectric Thickness (mils)	Dielectric Constant (DK) at 1 MHz	Width / Space / Width (mils)	Pair-to-Pair Space (mils)	Nominal Impedance (Ω)	Impedance Tolerance (+/-%)	Maximum Trace Length (inches) ¹
2.7	4.05	4 / 10 / 4	19	95 ²	17 ²	3.5
2.7	4.05	4 / 10 / 4	19	95 ²	15 ²	4
2.7	4.05	4 / 10 / 4	19	95	10	5
3.3	4.1	4.2 / 9 / 4.2	23	100 ²	17 ²	4
3.3	4.1	4.2 / 9 / 4.2	23	100	15	4.6
3.3	4.1	4.2 / 9 / 4.2	23	100	10	6
4	4.2	5 / 9 / 5	28	100 ²	17 ²	4.5
4	4.2	5 / 9 / 5	28	100	15	5.3
4	4.2	5 / 9 / 5	28	100	10	7

- Optimal MDI trace lengths should be maintained less than 3 inches on FR4 or FR408HR PCB material, longer MDI trace lengths can be achievable but might make it more difficult to achieve IEEE conformance. Simulations have shown deviations are possible if traces are kept short. Use cost considerations and stack up tolerance for differential pairs to determine length requirements.
- Deviations from 100 Ω nominal and/or tolerances greater than 15% decrease the maximum length for IEEE conformance.

Note: Use the MDI Differential Trace Calculator to determine the maximum MDI trace length for your trace geometry and board stack up. Contact your Intel representative for access.

The following factors can limit the maximum MDI differential trace lengths for IEEE conformance:

- Dielectric thickness
- Dielectric constant
- Nominal differential trace impedance
- Trace impedance tolerance
- Copper trace losses
- Additional devices, such as switches, in the MDI path might impact IEEE conformance.

Board geometry should also be factored in when setting trace length.

12.2.3.5 Ground Planes Under a Magnetics Module

The magnetic module chassis or output ground (secondary side of transformer) should be separated from the digital or input ground (primary side) by a physical separation of 80 mils minimum. Splitting the ground planes beneath the transformer minimizes noise coupling between the primary and secondary sides of the transformer and between the adjacent coils in the magnetics. This arrangement also improves the common mode choke functionality of magnetics module.

Integrated magnetics perform this ground plane separation internally. As a result, the ground plane split previously described is not needed when using integrated magnetics.

12.2.4 PHY MDI Lane Swap Configuration

The E610 provides flexible MDI LAN swaps for MDI board routing (see Figure 12-5) via an NVM setting.

- Default configuration - 0, 1, 2, 3 <-> A, B, C, D
- MDI swap configuration - 3, 2, 1, 0 <-> A, B, C, D

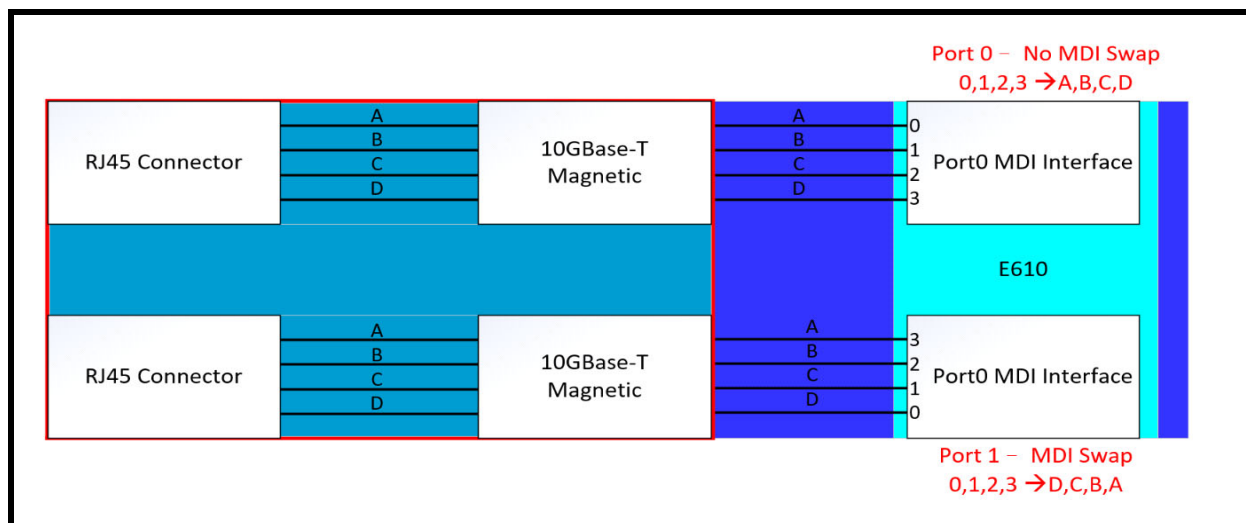


Figure 12-5. MDI Lane-Swap Configuration

12.2.5 Center Tap Connection Via Capacitors to Ground

The E610 has a voltage-mode driver. When using it in 10GBASE-T applications, it is required that a center tap be de-coupled to ground. When using an integrated magnetic, a 0.1 μ F capacitor should be connected from each center tap pin to ground. Similarly, when using a discrete magnetic, add a 0.1 μ F capacitor to each center tap pin for a total of four capacitors, as previously described.

12.3 Connecting the Power Supply Delivery Network

For the E610, it is necessary to provide an adequate power supply delivery network so that the device's performance meets expected performance requirements. To achieve this goal, both high speed and bulk decoupling capacitor networks are required in the design to ensure that frequency performance of the power supply delivery network is flat across the frequency spectrum. For details on how many high frequency and bulk decoupling capacitors are necessary to achieve this requirement along with the associated values of these capacitors, refer to the *E610 10GBASE-T Dual Port Ethernet Controller Reference Design* schematics.

To further ensure that these specific high frequency and bulk decoupling capacitors are placed in locations around and under the E610, Figure 12-6 breaks out the five required voltage supplies and color codes them according to power supply domain as a potential reference placement. Around the outside of the E610, highlighted bulk decoupling capacitors can be seen and because of their capacitance value and relative size, should be placed outside the BGA landing pattern. These decoupling capacitors still provide efficient bulk decoupling performance, as their capacitance value, along with the inductance of copper connecting them, still performs optimally.

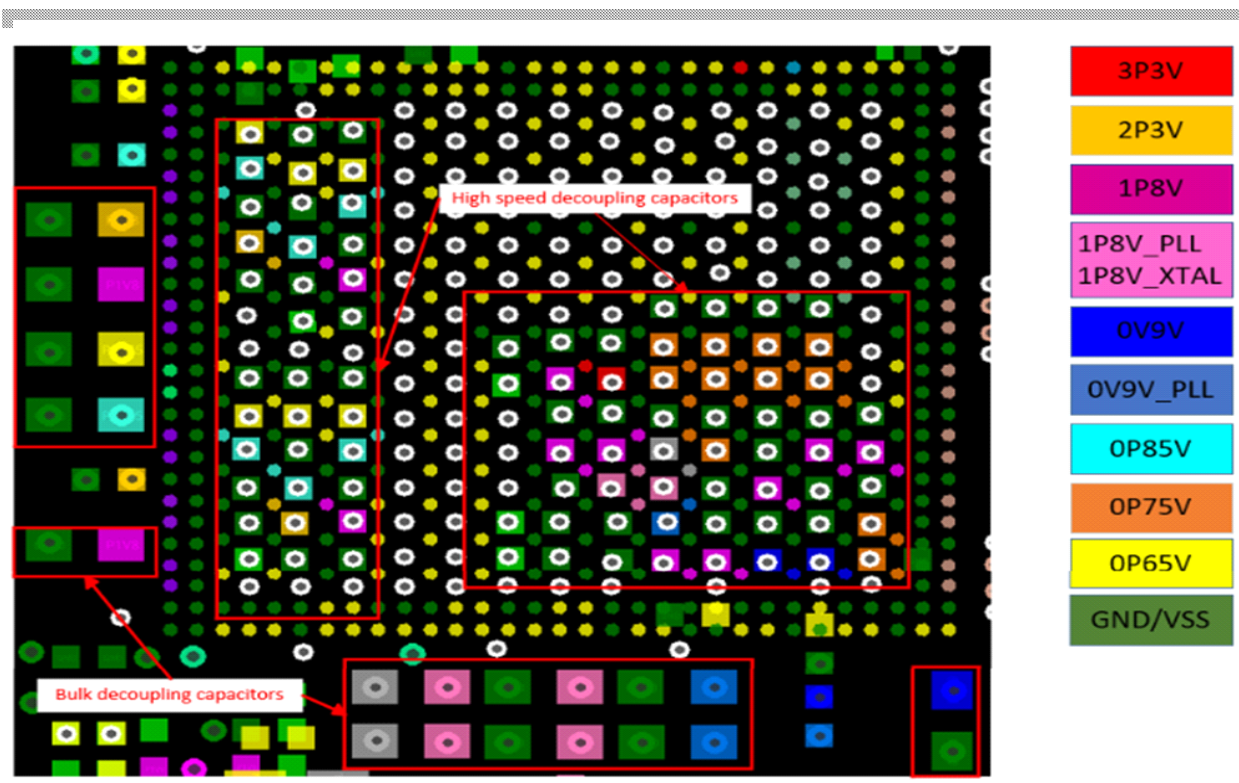


Figure 12-6. Power Supply Delivery Network

Within the BGA landing pattern, it is recommended that the high-speed decoupling capacitors be placed directly below the E610 on the secondary side of the PCB to ensure a low inductance path connection. For each power supply domain, the associated high speed decoupling capacitors should be placed in a way to provide coverage for all BGA locations. It is not Intel's recommendation that each power supply domain ball have a unique high-speed decoupling capacitor, but rather that the high-speed decoupling capacitors required from the reference schematic for each power supply domain provide consistent coverage across the targeted BGA connection locations.

12.4 Connecting the Flash Interface

12.4.1 Connecting the Flash

The E610 provides support for a Serial Peripheral Interface (SPI) Flash device that is made accessible to the system through the following:

- Flash Base Address register (PCIe Control register at offset 0x14 or 0x18).
- An address range of the IOADDR register, defined by the I/O Base Address register (PCIe Control register at offset 0x18 or 0x20).
- Expansion ROM Base Address register (PCIe Control register at offset 0x30)

12.4.2 Supported Flash Devices

The E610 uses SPI Flash. Flash is accessed automatically by the E610 after reset to provide pre-boot configuration data before it is accessed by host software. The remainder of the Flash space is available to software for storing the MAC address, serial numbers, and additional information. For a complete description of the content stored in Flash refer to [Section 6](#).

Supported Op Code:

- Write Enable (06)
- Read Status Register (05)
- Write Status Register (01)
- Read Data (03)
- Byte/Page Program (02) (program 1 to 256 data bytes)
- 4 KB Sector Erase (20)
- Chip Erase (C7)

Note: For the Write Status register op code, the written data is 0x00 (to cancel the default protection).

The E610 contains a 3.3V Flash I/O.

12.5 Connecting Manageability Interfaces

SMBus and NC-SI are optional interfaces for pass-through and/or configuration traffic between the MC and the E610. For a description of the operation mode of these interfaces, refer to [Section 7](#) and [Section 12](#).

12.5.1 Connecting the SMBus Interface

To connect the SMBus interface to the chipset or the MC; connect the SMBD, SMBCLK and SMBALRT_N signals to the corresponding pins of the chipset/MC. These pins require pull-up resistors to the 3.3V supply rail.

Note: If the interface is not used, the previously mentioned pull-up resistors on the SMBD, SMBCLK and SMBALRT_N signals still have to be in place.

It is recommended that the SMBus be connected to the chipset or MC for the Flash recovery solution. If the connection is to a MC, it is able to send the Flash release command.

12.5.2 Connecting the NC-SI Interface

The NC-SI interface is a connection to an external MC. It operates as a single interface with an external MC, where all traffic (other than header redirection) between the E610 and the MC flows through the interface.

12.5.2.1 External MC

The external MC is required to meet the electrical specifications called out in the NC-SI specification.

12.5.2.2 Single and Multi-Drop Applications

[Figure 12-7](#) shows the connectivity requirements and provides information about a single-drop application. This configuration only has a single package connected to the MC. The E610 does support a multi-drop NC-SI configuration architecture, including hardware arbitration support from the MC. refer to the NC-SI specification for requirements for multi-drop applications.

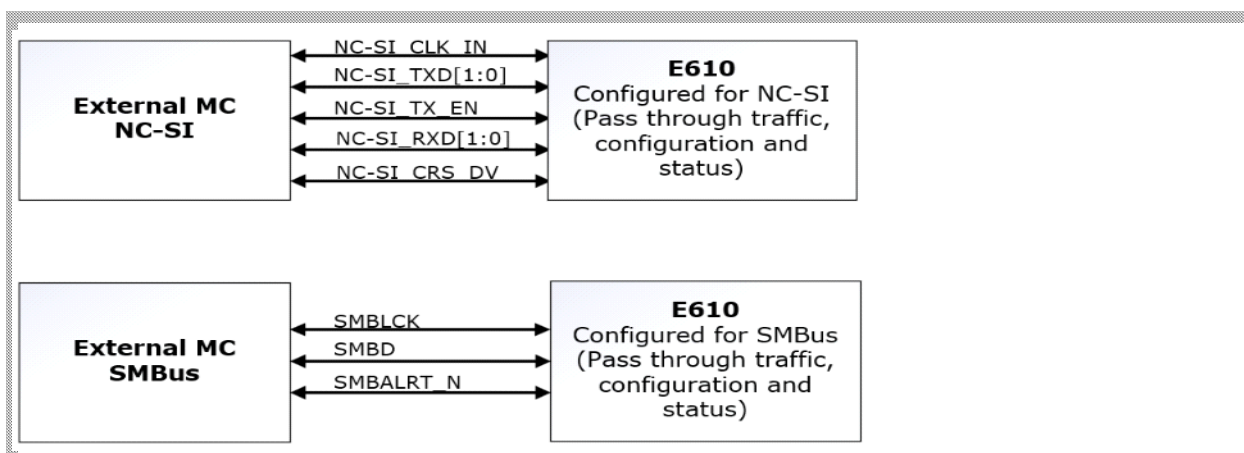


Figure 12-7. External MC Connections with NC-SI and SMBus

12.5.2.3 Pull-Ups and Pull-Downs

Depending on whether or not the NC-SI interface is used, different pull-up and pull-down resistors are required. Figure 12-8 shows the connectivity requirements necessary for interfacing a NC-SI compliant MC.

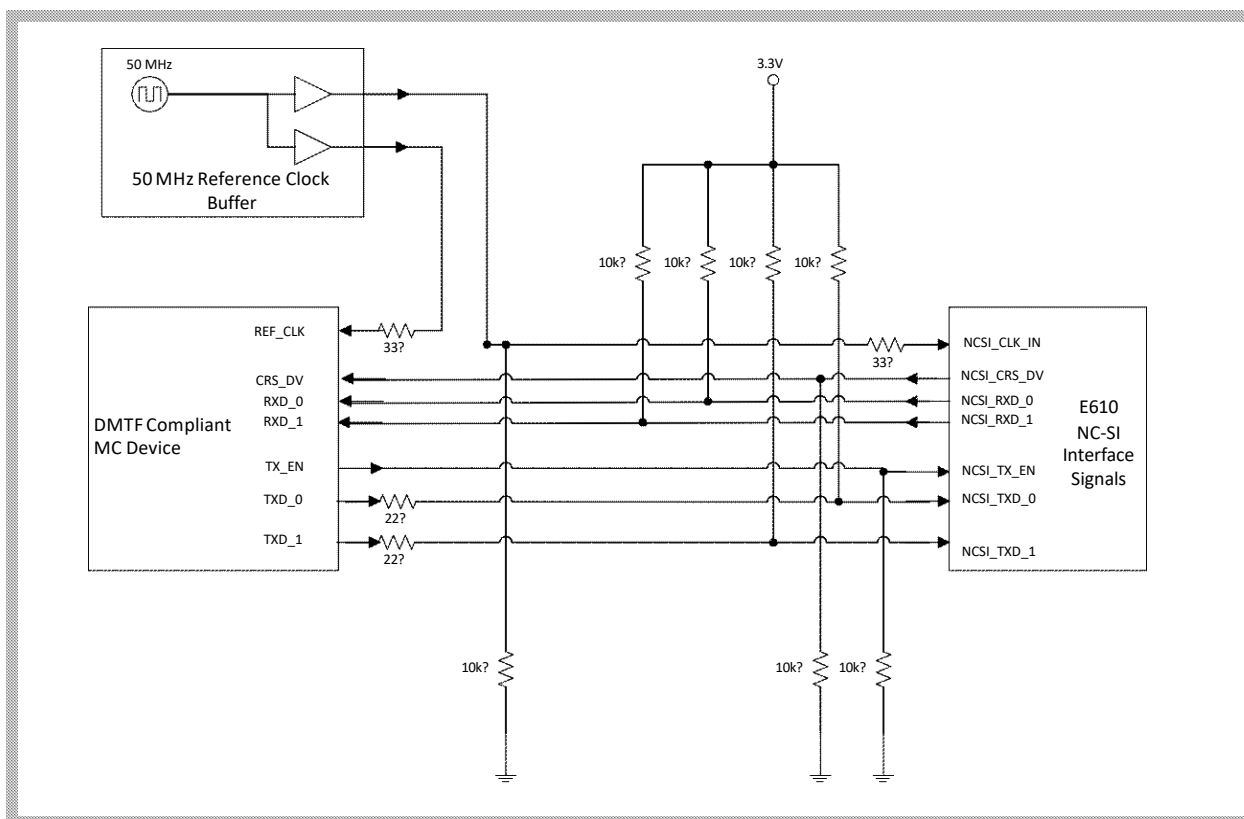


Figure 12-8. NC-SI Connections to an External MC

12.5.3 Layout Requirements

12.5.3.1 Board Impedance

The NC-SI manageability interface is a single ended signaling environment. It is recommended that a target board and trace impedance of 50 plus 20% and minus 10%. This ensures optimal signal integrity.

12.5.3.2 Trace Length Restrictions

It is recommended that a trace length maximum value from a board placement and routing topology perspective of eight inches for direct connect applications. This ensures signal integrity and quality is preserved from a design perspective and that compliance is met for NC-SI electrical requirements.

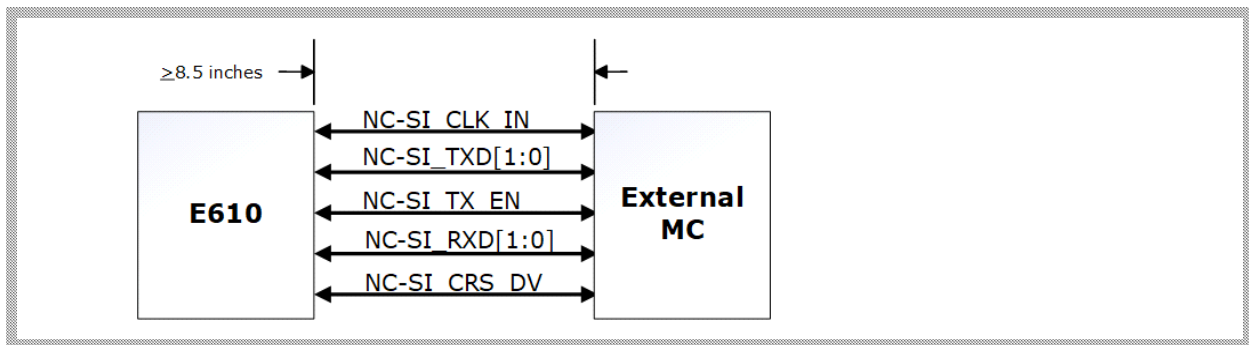


Figure 12-9. NC-SI Maximum Trace Length Requirement for Direct Connect Applications

For multi-drop applications, a spacing recommendation of a maximum of four inches between the two packages connected to the same MC. This is done to keep the overall length between the MC and the E610 within specification. See [Figure 12-11](#)

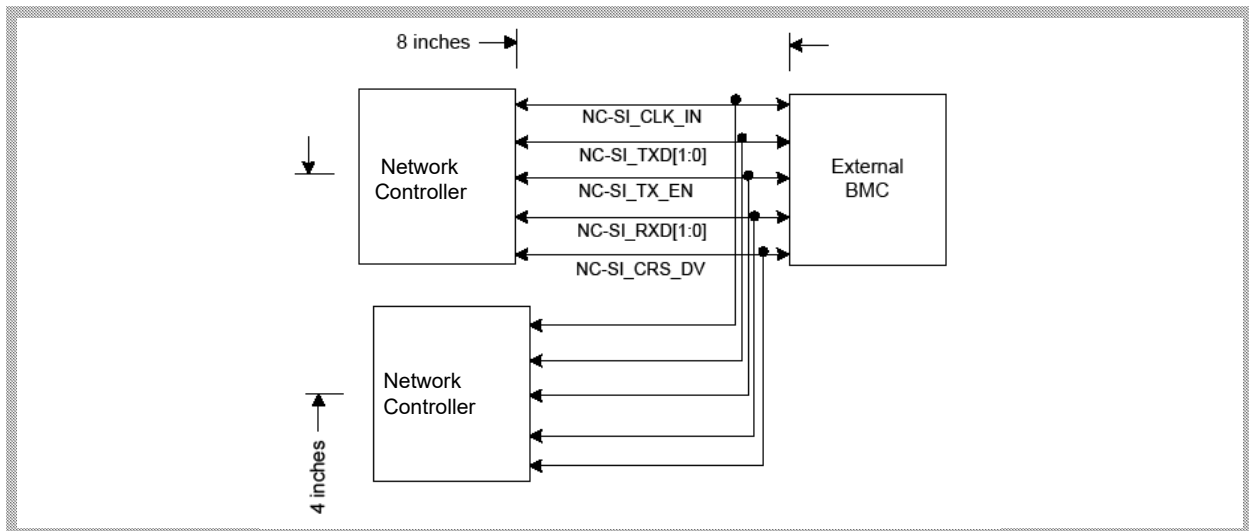


Figure 12-10. Spacing Recommendation for Multi-Drop Applications

12.5.3.3 Special Delay Requirements

To ensure that the E610 is able to communicate with a specification-compliant MC, the following guidelines must be followed:

- Total skew from the clock source to all devices should be less than 1 ns (less than or equal to an approximate 5.5 inch trace length skew).
- Each inch of trace has about 160 ps to 180 ps of delay, depending on the effective dielectric constant.

12.6 Connecting the Software-Definable Pins (SDPs)

The E610 has four SDPs per port (SDP16-SDP19 for port0; SDP20-SDP23 for port1) that can be used for miscellaneous hardware or software-controllable purposes. These pins and their function are bound to a specific LAN device. The pins can each be individually configured to act as either input or output pins via Flash. The initial value in the case of an output can also be configured in the same way. However, the E610 default for any of these pins is to be configured as outputs.

To avoid signal contention, all four pins (per port) are set as input pins until the Flash configuration has been loaded.

Choose the correct SDPs for specific applications. All pins are tri-state buffers. Consider that four of these pins per port can be used to support IEEE1588 auxiliary devices, input for external interrupts, indication for thermal-sensor event or as general-purpose interrupt (GPI) inputs. To act as GPI pins, the desired pins must be configured as inputs. A separate GPI interrupt-detection enable is then used to enable rising-edge detection of the input pin (rising-edge detection occurs by comparing values sampled at 62.5 MHz, as opposed to an edge-detection circuit). When detected, a corresponding GPI interrupt is indicated in the Interrupt Cause register.

When connecting the SDPs to different digital signals please keep in mind that these are 3.3V signals and use level shifting if necessary.

In default setting with NVM, two of four SDP pins are assigned to IEEE1588 function for each port (SDP16-SDP17 for Port0, SDP22-SDP23 for Port1), use SDP18-SDP21 for software-controllable purposes.

The use, direction, and values of SDPs are controlled and accessed using fields in the Extended SDP Control (ESDP) and Extended OD SDP Control (EODSDP) registers.

12.7 Connecting the Light Emitting Diodes (LEDs)

The E610 provides three programmable high-current sink (active low) outputs per port to directly drive LEDs for link activity and speed indication. Each LAN device provides an independent set of LED outputs; these pins and their function are bound to a specific LAN device. Each of the three LED outputs can be configured to select the event, state, or activity indicated on that output. In addition, each LED can be individually configured for output polarity, as well as for blinking versus non-blinking (steady state) indication.

The LED ports are fully programmable through the Flash interface (LEDCTL register). In addition, the hardware-default configuration for all LED outputs can be specified via a Flash field, thus supporting LED displays configurable to a particular OEM preference.

Provide separate current limiting resistors for each LED connected.

Since the LEDs are likely to be placed close to the board edge and to external interconnect, take care to route the LED traces away from potential sources of EMI noise. In some cases, it might be desirable to attach filter capacitors.

12.8 Connecting Miscellaneous Signals

12.8.1 LAN Disable

For a LAN on Motherboard (LOM) design, it might be desirable for the system to provide BIOS setup capability for selectively enabling or disabling E610 PCI devices, PCI functions, or external Ethernet ports and the associated PCI function. It provides end with more control over system resource management and avoids conflicts with add-in NIC solutions. E610 provides support for selectively enabling or disabling one or more LAN PCI device(s) in the system.

E610 provides options to disable its external Ethernet ports and/or PCI functions:

- A strapping pin (DEV_DIS_N) is sampled on RSMRST# and PCIe resets to disable Ethernet ports. The specific port(s) to be disabled is determined from NVM. Additional NVM configuration is provided to determine which PCI functions are disabled at the same time. The expected usage is to disable the PCI functions associated with the disabled ports.
- A strapping pin (PCI_DIS_N) is sampled on RSMRST# and PCIe resets to disable PCI functions. The specific functions that are disabled are determined from the NVM.

Some guidelines on usage of these straps:

- During power-up, the PCI_DIS_N and DEV_DIS_N pins are ignored until the NVM is read. From that point, E610 might disable all PCI functions if either PCI_DIS_N or DEV_DIS_N is asserted.
- De-assertion of the PCI_DIS_N or DEV_DIS_N pins requires the system to issue a power-on reset/ RSMRST#/PE_RST_N to E610 to re-enable any disabled PCI functions or external Ethernet ports.
- The PCI_DIS_N and DEV_DIS_N pins should maintain their state during system reset and system sleep states. It should also ensure the proper default value on system power up. For example, one could use a GPIO pin that defaults to 1b (enable) and is on system suspend power (such that it maintains its state in S0-S5 ACPI states).

When function 0 is disabled then it does not disappear from the PCIe configuration space. Rather, the function presents itself as a dummy function. The device ID and class code of this function changes to other values (dummy function device ID 0x10A6, class code 0xFF0000). In addition, the function does not require any I/O space and does not require an interrupt line. It requires a minimal memory space (4 KB) that is not mapped to internal registers.

Mapping between function and LAN ports is listed in the following table.

Table 12-3. PCI Functions Mapping

PCI Function #	LAN Function Select	Function 0	Function 1
Both LAN functions are enabled	0	LAN 0	LAN 1
	1	LAN 1	LAN 0
LAN 0 is disabled	0	Dummy	LAN 1
	1	LAN 1	Disable
LAN 1 is disabled	0	LAN 0	Disable
	1	Dummy	LAN 0
Both LAN functions are disabled	Both PCI functions are disabled. The E610 is in low power mode.		

The following rules apply to function disable:

- When function 0 is disabled, it is converted into a dummy PCI function. Function 1 is not affected.
- When function 1 is disabled, it disappears from the PCI configuration space.
- The disabled LAN port is still available for manageability purposes through the NVM mechanism.
- The disabled LAN port is still available for manageability purposes if disabled through the NVM. In this case, and if the *LPLU* bit is set, the PHY firmware attempts to create a link at the lowest supported speed. The disabled LAN port is not available for manageability purposes if disabled through the strapping mechanism.
- Dummy function mode should not be used in PCI IOV mode (since PF0 is required to support certain functionality).

The following NVM bit control function disable:

- The LAN_FUNCTION_SEL field in NVM (reflected in FACTPS.LAN_FUNCTION_SEL bit) defines the correspondence between LAN port and PCI function.

When a particular LAN is fully disabled (Ethernet port with associated PF), all internal clocks to that LAN are disabled, the device is held in reset, and the internal PHY for that LAN is powered down. In both modes, the device does not respond to PCI configuration cycles. Effectively, the LAN device becomes invisible to the system from both a configuration and power consumption standpoint.

Table 12-4. PCI Functions Mapping (Dummy Function Mode)

PCI Function #	LAN Function Select	Function 0	Function 1
Both LAN functions are enabled	0	LAN 0	LAN 1
	1	LAN 1	LAN 0
LAN 0 is disabled	0	Dummy	LAN 1
	1	LAN 1	Disable
LAN 1 is disabled	0	LAN 0	Disable
	1	Dummy	LAN 0
Both LAN functions are disabled	Both PCI functions are disabled. The E610 is in low power mode.		

When both LAN ports are disabled following a power on reset / LAN_PWR_GOOD / PE_RST_N / in-band reset, the LAN_DIS_N signals should be tied statically to low. At this state, the E610 is disabled, LAN ports are powered down, all internal clocks are shut and the PCIe connection is powered down (similar to L2 state).

12.8.2 Event Flow for Enable/Disable Functions (HII)

This section describes the expected flow to disable or enable PCI functions or Ethernet ports. Following a power-on reset/RSMRST#/PE_RST_N/in-band reset, the DEV_DIS_N and PCI_DIS_N signals should be driven high (or left unconnected) for normal operation.

The following example assumes that PCI functions and/or Ethernet ports are being disabled during the BIOS initialization phase:

1. Following a power-up sequence, the DEV_DIS_N and PCI_DIS_N signals are driven high.
2. The PCIe link is established following PE_RST_N.
3. BIOS goes through PCI bus enumeration.
4. BIOS recognizes that the PCI functions in E610 should be disabled.
5. The BIOS drives the DEV_DIS_N or PCI_DIS_N signal to a low level.
6. The BIOS asserts PCIe reset, either in-band or via PE_RST_N.
7. As a result, E610 samples the DEV_DIS_N and PCI_DIS_N signals and disables the PCI functions and/or external Ethernet ports.
8. BIOS might do device enumeration a second time (the disabled PCI functions are invisible or changed to dummy function).
9. Proceed with normal operation.
10. Re-enable could be done by driving the DEV_DIS_N and PCI_DIS_N signals high and re-issuing a power-on reset/RSMRST#/PE_RST_N/in-band reset.

12.9 Connecting the JTAG Port

The E610 contains a test access port (3.3 V only) conforming to the IEEE 1149.1-2001 Edition (JTAG) specification. To use the test access port, connect these balls to pads accessible by specific test equipment.

For proper operation, a pull-down resistor should be connected to the JTCK and J_RST_N signals and pull up resistors to the JTDO, JTMS and JTDI signals.

A Boundary Scan Definition Language (BSDL) file describing the E610 is available for use in specific test environments.

12.10 Power On Reset (POR)

After power is applied, the E610 must be reset. There are two ways to do this:

1. Using the internal Power on Reset (POR) circuit with external LAN_PWR_GOOD signal pulled up.
2. Using the internal POR circuit with the external reset as LAN_PWR_GOOD signal generated by the system.

The power supply sequencing timing requirement between the 3.3V and 2.1V, 1.2V, and 0.83V power rails has to be met (see [Section 12.3.8](#)).

There are two options for implementing POR:

1. Internal POR (recommended as default).
 - a. BYPASS_POR signal (pull down).

- b. LAN_PWR_GOOD signal (pull up).
- 2. External reset with LAN_PWR_GOOD signal:
 - a. BYPASS_POR signal (pull down).
 - b. LAN_PWR_GOOD signal generated by the system following device power-up timing.

12.11 Clock Source Design Considerations

This section provides information regarding crystals for use with the E610.

All designs require an external clock. The only option for this clock source is a 50 MHz external crystal. The E610 uses the crystal to generate clocks for the high speed interfaces.

The chosen crystal vendor should be consulted early in the design cycle. Crystal manufacturers familiar with networking equipment clock requirements might provide assistance in selecting an optimum, low-cost solution.

12.11.1 Quartz Crystal

Quartz crystals are the mainstay of frequency components due to low cost and ease of implementation. They are available from numerous vendors in many package types with various specification options.

12.11.2 Vibrational Mode

Crystals are available in both fundamental and third overtone. Unless there is a special need for third overtone, use fundamental mode crystals. At any operating frequency, third overtone crystals are thicker and more rugged than fundamental mode crystals. Third overtone crystals are more suitable for use in military or harsh industrial environments. Third overtone crystals require a trap circuit (extra capacitor and inductor) in the load circuitry to suppress fundamental mode oscillation as the circuit powers up. Selecting values for these components is beyond the scope of this document.

12.11.3 Frequency Tolerance

The frequency tolerance for an Ethernet Platform LAN Connect is dictated by the IEEE 802.3 specification as ± 50 parts per million (ppm). This measurement is referenced to a standard temperature of 25° C. It is recommended that a frequency tolerance of ± 10 ppm be used.

12.11.4 Temperature Stability and Environmental Requirements

Temperature stability is a standard measure of how the oscillation frequency varies over the full operational temperature range (and beyond). Several optional temperature ranges are currently available, including -10 °C to +70 °C for industrial environments. Some vendors separate operating temperatures from temperature stability. Manufacturers can also list temperature stability as 50 ppm in their data sheets.

Note: Crystals also carry other specifications for storage temperature, shock resistance, and reflow solder conditions. Crystal vendors should be consulted early in the design cycle to discuss the application and its environmental requirements.

12.11.5 Calibration Mode

The terms series-resonant and parallel-resonant are used to describe crystal oscillator circuits. Specifying parallel mode is critical to determining how the crystal frequency is calibrated at the factory. A crystal specified and tested as series resonant oscillates without problem in a parallel-resonant circuit, but the frequency is higher than nominal by several hundred parts per million. The purpose of adding load capacitors to a crystal oscillator circuit is to establish resonance at a frequency higher than the crystal's inherent series resonant frequency.

12.11.6 Reference Crystal Circuit

Figure 12-11 illustrates a simplified schematic of the internal oscillator circuit. Pin X1 and X2 refers to XTAL1 and XTAL2 in the E610, respectively. The crystal and the capacitors form a feedback element for the internal inverting amplifier. This combination is called parallel-resonant, because it has positive reactance at the selected frequency. In other words, the crystal behaves like an inductor in a parallel LC circuit. Oscillators with piezoelectric feedback elements are also known as Pierce oscillators.

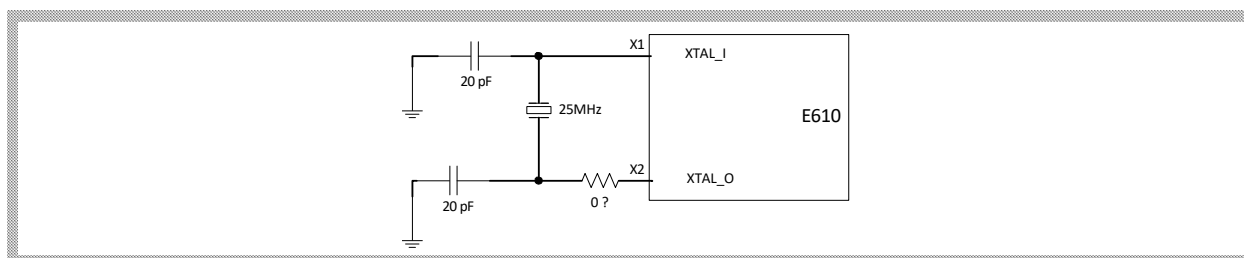


Figure 12-11. Oscillator Circuit

12.11.7 Crystal Load Capacitance

The formula for crystal load capacitance is as follows:

$$C_L = \frac{(C1 \cdot C2)}{(C1 + C2)} + C_{stray}$$

where $C1 = C2 = 18 \text{ pF}$ and C_{stray} = allowance for additional capacitance in pads, traces and the chip carrier within the Ethernet device package. An allowance of 3 pF to 7 pF accounts for lumped stray capacitance. The calculated load capacitance is 16 pF with an estimated stray capacitance of about 5 pF. Individual stray capacitance components can be estimated and added. For example, surface mount pads for the load capacitors add approximately 2.5 pF in parallel to each capacitor. This technique is especially useful if Y1, C1 and C2 must be placed farther than approximately one-half (0.5) inch from the device. Thin circuit boards generally have higher stray capacitance than thick circuit boards. Oscillator frequency should be measured with a precision frequency counter where possible. The load

specification or values of C1 and C2 should be fine-tuned for the design. As the actual capacitance load increases, the oscillator frequency decreases.

Note: C1 and C2 can vary by as much as 5% (approximately 1 pF) from their nominal values.

12.11.8 Shunt Capacitance

The shunt capacitance parameter is relatively unimportant compared to load capacitance. Shunt capacitance represents the effect of the crystal's mechanical holder and contacts. The shunt capacitance should equal a maximum of 5 pF.

12.11.9 Equivalent Series Resistance (ESR)

ESR is the actual component of the crystal's impedance at the calibration frequency, which the inverting amplifier's loop gain must overcome. ESR varies inversely with frequency for a given crystal family. The lower the ESR, the faster the crystal starts up. Use crystals with an ESR value of 50 Ω or better.

12.11.10 Driver Level

Drive level refers to power dissipation in use. The allowable drive level for a Surface Mounted Technology (SMT) crystal is less than its through-hole counterpart, because surface mount crystals are typically made from narrow, rectangular AT strips, rather than circular AT quartz blanks. Some crystal data sheets list crystals with a maximum drive level of 1 mW. However, Intel Ethernet controllers drive crystals to a level less than the suggested 0.3 mW value. This parameter does not have much value for on-chip oscillator use.

12.11.11 Aging

Aging is a permanent change in frequency (and resistance) occurring over time. This parameter is most important in its first year because new crystals age faster than old crystals. Use crystals with a maximum of ± 5 ppm per year aging.

12.11.12 Reference Crystal

The normal tolerances of the discrete crystal components can contribute to small frequency offsets with respect to the target center frequency. To minimize the risk of tolerance-caused frequency offsets causing a small percentage of production line units to be outside of the acceptable frequency range, it is important to account for those shifts while empirically determining the proper values for the discrete loading capacitors, C1 and C2. Even with a perfect support circuit, most crystals oscillate slightly higher or slightly lower than the exact center of the target frequency. Therefore, frequency measurements (which determine the correct value for C1 and C2) should be performed with an ideal reference crystal. When the capacitive load is exactly equal to the crystal's load rating, an ideal reference crystal is perfectly centered at the desired target frequency.

12.11.13 Reference Crystal Selection

There are several methods available for choosing the appropriate reference crystal. These are listed below.

- If a Saunders and Associates (S&A) crystal network analyzer is available, discrete crystal components can be tested until one is found with zero or nearly zero ppm deviation (with the appropriate capacitive load). A crystal with zero or near zero ppm deviation is a good reference crystal to use in subsequent frequency tests to determine the best values for C1 and C2.
- If a crystal analyzer is not available, the selection of a reference crystal can be done by measuring a statistically valid sample population of crystals, which has units from multiple lots and approved vendors. The crystal, which has an oscillation frequency closest to the center of the distribution, should be the reference crystal used during testing to determine the best values for C1 and C2.
- It might also be possible to ask the approved crystal vendors or manufacturers to provide a reference crystal with zero or nearly zero deviation from the specified frequency when it has the specified CLoad capacitance.

When choosing a crystal, keep in mind that to comply with IEEE specifications for 100/ 1000/10GBase-T Ethernet LAN, the transmitter reference frequency must be precise within ± 50 ppm. Intel® recommends using a transmitter reference frequency that is accurate to within ± 10 ppm to account for variations in crystal accuracy due to crystal manufacturing tolerance.

12.11.14 Circuit Board

Since dielectric layers of the circuit board are allowed some reasonable variation in thickness, stray capacitance from the printed board (to the crystal circuit) also vary. If the thickness tolerance for the outer layers of dielectric are controlled within ± 17 percent of nominal, the circuit board should not cause more than ± 2 pF variation to the stray capacitance at the crystal. When tuning crystal frequency, it is recommended that at least three circuit boards are tested for frequency. These boards should be from different production lots of bare circuit boards. Alternatively, a larger sample population of circuit boards can be used. A larger population increases the probability of obtaining the full range of possible variations in dielectric thickness and the full range of variation in stray capacitance. Next, the exact same crystal and discrete load capacitors (C1 and C2) must be soldered onto each board and the LAN reference frequency should be measured on each circuit board. The circuit board, which has a LAN reference frequency closest to the center of the frequency distribution, should be used while performing the frequency measurements to select the appropriate value for C1 and C2.

12.11.15 Temperature Changes

Temperature changes can cause crystal frequency to shift. Frequency measurements should be done in the final system chassis across the system's rated operating temperature range.

12.12 PCB Guidelines

This section describes the general PCB design guidance targeted as supplementary information in addition to the specific requirements and recommendations for individual interfaces. For items not directly covered in the specific interface sections, these guidance recommendations apply to the design.

12.12.1 Board Stack-Up Example

PCBs for these designs typically have six, eight, or more layers. Although the E610 does not dictate stackup, the following examples show typical stack-up options.

Microstrip Example:

- Layer 1 is a signal layer.
- Layer 2 is a ground layer.
- Layer 3 is used for power planes.
- Layer 4 is a signal layer. Careful routing is necessary to prevent crosstalk with layer 5.
- Layer 5 is a signal layer. Careful routing is necessary to prevent crosstalk with layer 4.
- Layer 6 is used for power planes.
- Layer 7 is a signal ground layer.
- Layer 8 is a signal layer.

Note: Layers 4 and 5 should be used mostly for low-speed signals because they are referenced to potentially noisy power planes that might also be slotted.

Stripline Example:

- Layer 1 is a signal layer.
- Layer 2 is a ground layer.
- Layer 3 is a signal layer.
- Layer 4 is used for power planes.
- Layer 5 is used for power planes.
- Layer 6 is a signal layer.
- Layer 7 is a signal ground layer.
- Layer 8 is a signal layer.

Note: To avoid the effect of the potentially noisy power planes on the high-speed signals, use offset stripline topology. The dielectric distance between the power plane and signal layer should be three times the distance between ground and signal layer.

This board stack-up configuration can be adjusted to conform to company-specific design rules.

12.12.2 Customer Reference Board Stack-Up Example

Layer		Cu Weight	Proposed Thickness (mils)	Structure (Prepreg glass cloth & resin content%. Core construction)
	Solder mask		0.40	
L1	Top	1/3oz+Plating	1.38	HTE
	Prepreg		2.90	1x1078 MR
L2	GND	Hoz	0.63	RTF
	Core		5	core 1x2112
L3	Signal	Hoz	0.63	RTF
	Prepreg		5.01	1x1078 HR
L4	GND	1oz	1.22	RTF
	Core		28	core 2x2116
L5	Signal	1oz	1.22	RTF
	Prepreg		5.01	1x1078 HR
L6	PWR	Hoz	0.63	RTF
	Core		5	core 2x2116
L7	PWR	Hoz	0.63	RTF
	Prepreg		2.9	1x1078 R
L8	Bottom	1/3oz+Plating	1.38	HTE
	Solder mask		0.40	
Finished Thickness (mils)			61.5mils (+/-6mils)	

Following example shows impedance control over above recommended 8 layers PCB stackup ([Table 1](#)).

Table Impedance Control

Layer	Required Impedance (Ohms)	Calculated Impedance (Ohms)	Tolerance (Ohms)	Impedance Type	Reference Layers Upper / Lower	Design L/W (mils)	Design Space (mils)
L1	100.0	99.65	±10.0	diff coated microstrip	None / L2	3.94	7.28
L1	50.0	50.03	±5.0	se coated microstrip	None / L2	4.72	---
L3	100.0	99.65	±10.0	diff stripline	L2 / L4	3.5	5
L3	50.0	49.68	±5.0	se stripline	L2 / L4	4.45	---
L6	100.0	99.62	±10.0	diff stripline	L5 / L7	3.5	5
L6	50.0	49.68	±5.0	se stripline	L5 / L7	4.45	---
L8	100.0	99.65	±10.0	diff coated microstrip	L7 / None	3.94	7.28
L8	50.0	50.03	±5.0	se coated microstrip	L7 / None	4.72	---

12.12.3 Via Usage

Use vias to optimize signal integrity. Figure 12-12 shows correct via usage. Figure 12-13 shows the type of topology that should be avoided.

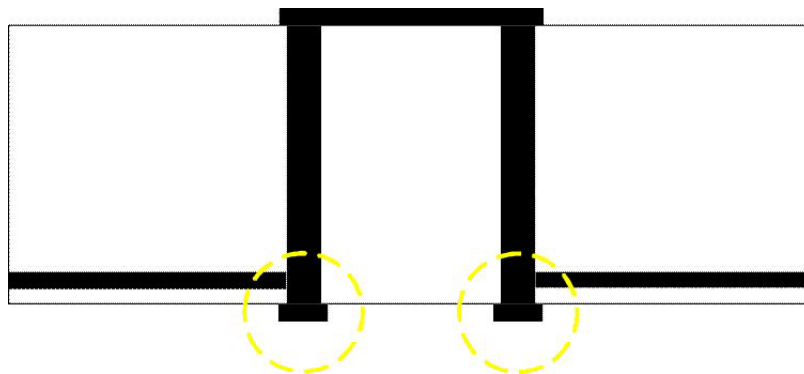


Figure 12-12. Correct Via Usage

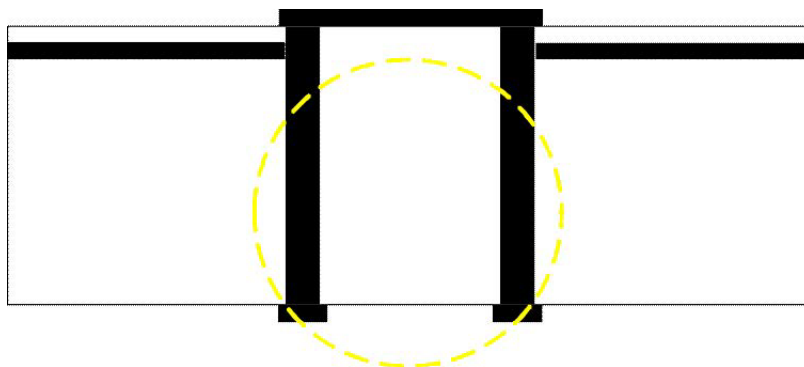


Figure 12-13. Incorrect Via Usage

Any via stubs on the MDI differential signal traces must be less than 45 mils in length. Keeping MDI signal via stubs less than or equal to 20 mils is preferable.

Place ground vias adjacent to signal vias used for the MDI interface. DO NOT embed vias between the high-speed signals but place them adjacent to the signal vias. This helps to create a better ground path for the return current of the AC signals, which also helps address impedance mismatches and EMC performance.

It is recommended that, in the breakout region between the vias and the capacitor pad, target a Z0 for the via to capacitor trace equal to 50 Ω . This minimizes impedance imbalance.

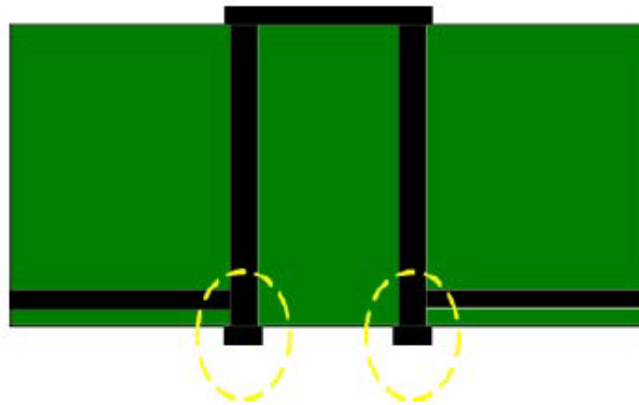


Figure 12-14. Incorrect Via Usage for Differential Pair

12.12.4 Reference Planes

Do not cross plane splits with the MDI high-speed differential signals. This causes impedance mismatches and negatively affects the return current paths for the board design and layout. See [Figure 12-15](#).

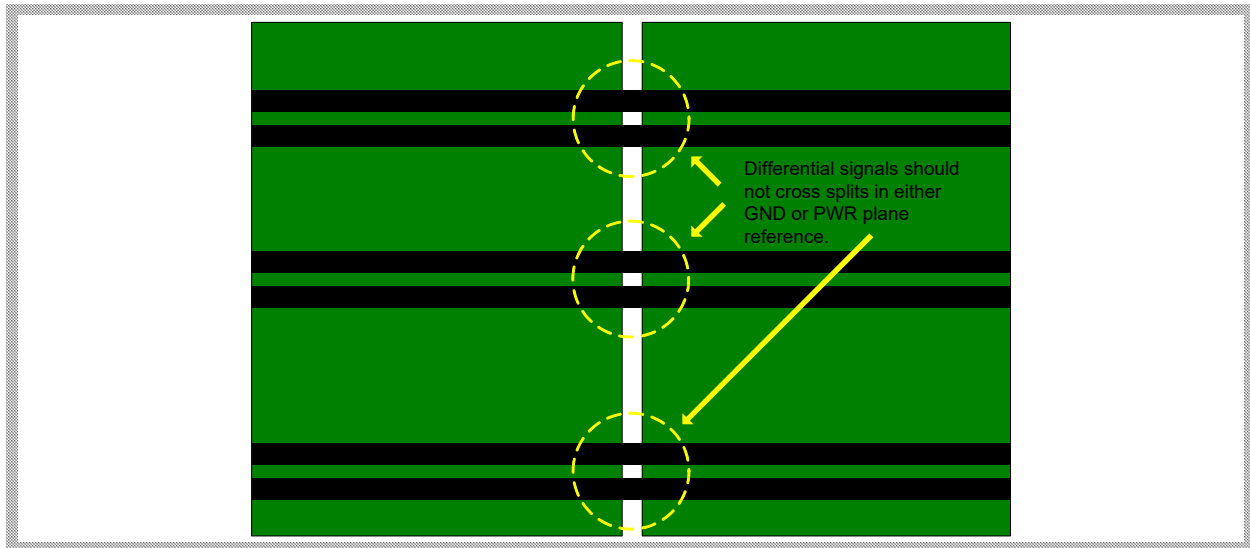


Figure 12-15. Improper Differential Signal Routing - Plane Split

Traces should not cross power or ground plane splits if at all possible. Traces should stay seven times the dielectric height away from plane splits or voids. If traces must cross splits, capacitive coupling should be added to stitch the two planes together to provide a better AC return path for the high-speed signals. To be effective, the capacitors should have low ESR and low equivalent series inductance.

Note: Even with plane split stitching capacitors, crossing plane splits is extremely high risk for 10GBASE-T designs.

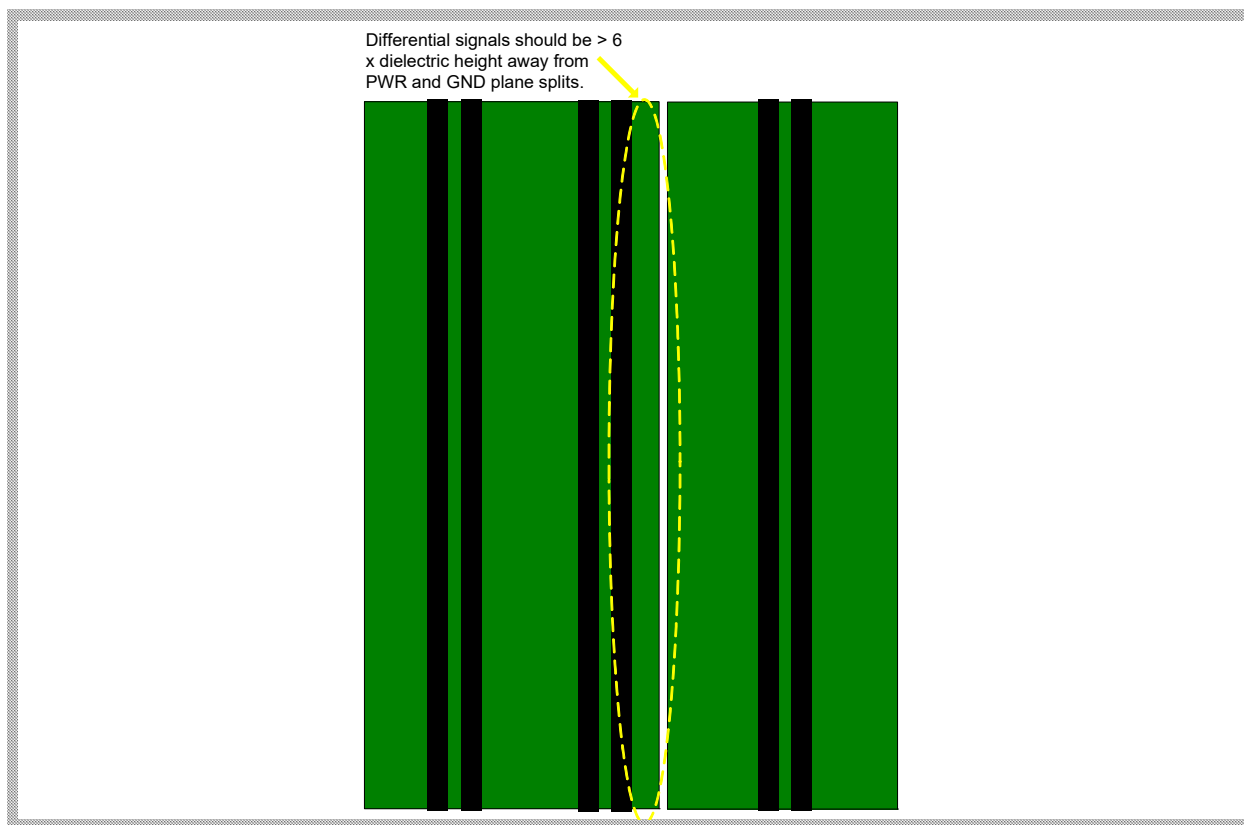


Figure 12-16. Differential Signal Routing - Plane split and Void Proximity

It is recommended that the MDI signals stay at least seven times the dielectric height away from any power or ground plane split. This improves impedance balance and returns current paths.

If a high-speed signal needs to reference a power plane, ensure that the height of the secondary (power) reference plane is at least 3 x the height of the primary (ground) reference plane.

12.12.5 Reducing Circuit Inductance

Traces should be routed over a continuous reference plane with no interruptions. If there are vacant areas on a reference or power plane, the signal conductors should not cross the vacant area. Routing over a void in the reference plane causes impedance mismatches and usually increases radiated noise levels. Noisy logic grounds should NOT be located near or under high-speed signals or near sensitive analog pin regions of the LAN silicon. If a noisy ground area must be near these sensitive signals or IC pins, ensure sufficient decoupling and bulk capacitance in these areas. Noisy logic and switching power supply grounds can sometimes affect sensitive DC subsystems such as analog to digital conversion, operational amplifiers, etc.

All ground vias should be connected to every ground plane; and similarly, every power via should be equally potential power planes. This helps reduce circuit inductance. Another recommendation is to physically locate grounds to minimize the loop area between a signal path and its return path. Rise and fall times should be as slow as possible while still meeting the relevant electrical requirements because signals with fast rise and fall times contain many high frequency harmonics, which can radiate significantly. The most sensitive signal returns closest to the chassis ground should be connected together. This results in a smaller loop area and reduces the likelihood of crosstalk. The effect of different configurations on the amount of crosstalk can be studied using electronics modeling and simulation software.

12.12.6 Signal Isolation

To maintain the best signal integrity, keep digital signals far away from the analog traces. A good rule to follow is no digital signal should be within 7x to 10x dielectric height of the differential pairs. If digital signals on other board layers cannot be separated by a ground plane, they should be routed at a right angle (90 degrees) to the differential signal traces. If there is another Ethernet controller on the board, take care to keep the differential pairs away from that circuit. The same thing applies to switching regulator traces.

Rules to follow for signal isolation:

- Separate and group signals by function on separate board layers if possible. Maintain a separation that is at least seven times the thinnest adjacent dielectric height between all differential pairs (Ethernet) and other nets, but group associated differential pairs together.
- Over the length of the trace run, each differential pair should be at least seven times the thinnest adjacent dielectric height away from any parallel signal traces.
- Physically group together all components associated with one clock trace to reduce trace length and radiation.
- Isolate other I/O signals from high-speed signals to minimize crosstalk. Crosstalk can increase radiated EMI and can also increase susceptibility to EMI from other signals.
- Avoid routing high-speed LAN traces near other high-frequency signals associated with a video controller, cache controller, processor, or other similar devices.

12.12.7 Traces for Decoupling Capacitors

Traces between decoupling and I/O filter capacitors should be as short and wide as practical. Long and thin traces are more inductive and reduce the intended effect of decoupling capacitors. Also, for similar reasons, traces to I/O signals and signal terminations should be as short as possible. Vias to the decoupling capacitors should be sufficiently large in diameter to decrease series inductance.

12.12.8 Power and Ground Planes

Good grounding requires minimizing inductance levels in the interconnections and keeping ground returns short, signal loop areas small, and locating decoupling capacitors at or near power inputs to bypass to the signal return. This significantly reduces EMI radiation.

These guidelines reduce circuit inductance in NICs and LOMs:

- Route traces over a continuous plane with no interruptions. Do not route over a split power or ground plane. If there are vacant areas on a ground or power plane, avoid routing signals over the vacant area. Routing signals over power or ground voids increases inductance and increases radiated EMI levels.
- Use distance and/or extra decoupling capacitors to separate noisy digital grounds from analog grounds to reduce coupling. Noisy digital grounds can affect sensitive DC subsystems.
- All ground vias should be connected to every ground plane, and every power via should be connected to all power planes at equal potential. This helps reduce circuit inductance.
- Physically locate grounds between a signal path and its return. This minimizes the loop area.
- Avoid fast rise/fall times as much as possible. Signals with fast rise and fall times contain many high frequency harmonics, which can radiate EMI.
- Do not route high-speed signals near switching regulator circuits.
- It's acceptable to put ground fill or thieving on the trace layers, but preferably not closer than 50 mils to the differential traces and the connector pins.
- If differential traces must be routed on another layer, the signal vias should carry the signal to the opposite side of the PCB (to be near the top of the PCB), AND if the high-speed signals are being routed between two connectors on the same board, before the signal traces reach the second connector, they must return to the original signal layer (before reaching the connector pin). This strategy keeps via stubs short without requiring back drilling.
- Each time differential traces make a layer transition (pass through a pair of signal vias), there must be at least one ground via located near each signal via. Two ground vias near each signal via is better. See [Figure 12-17](#) and [Figure 12-18](#).

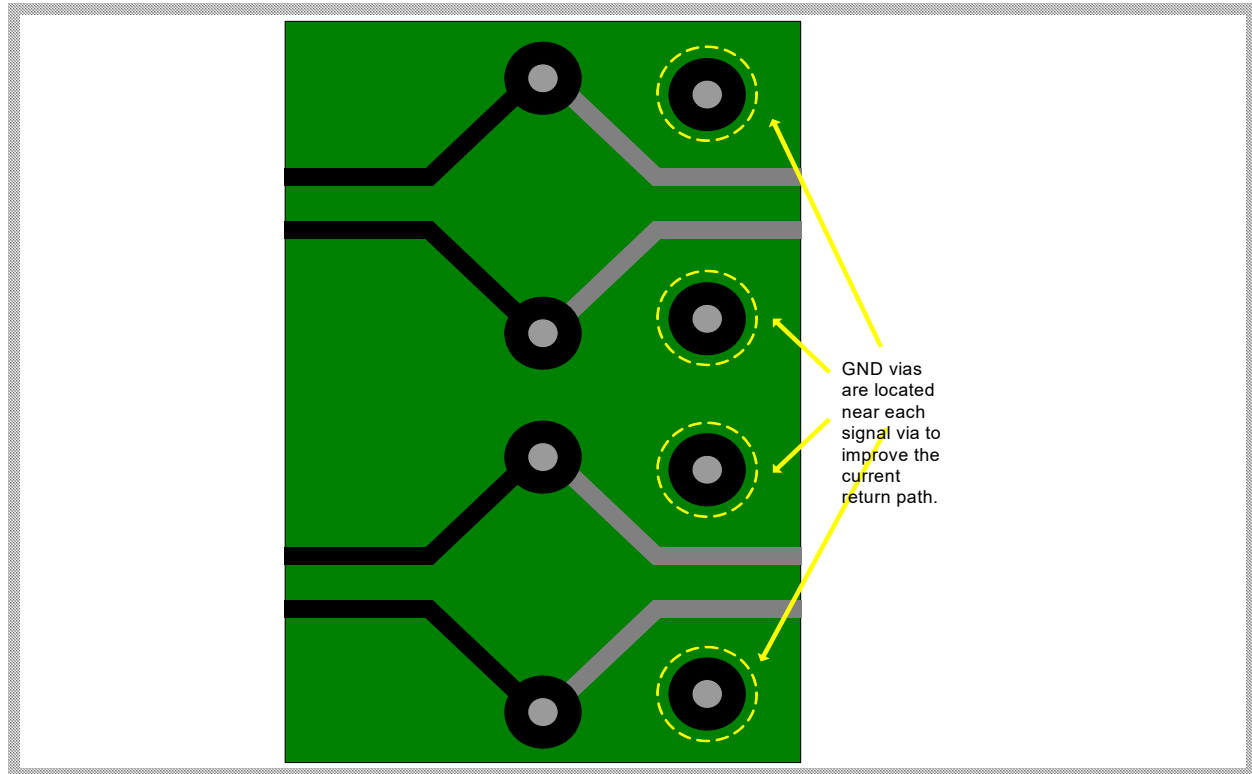


Figure 12-17. Return Path Vias for Differential Signals - Acceptable Example

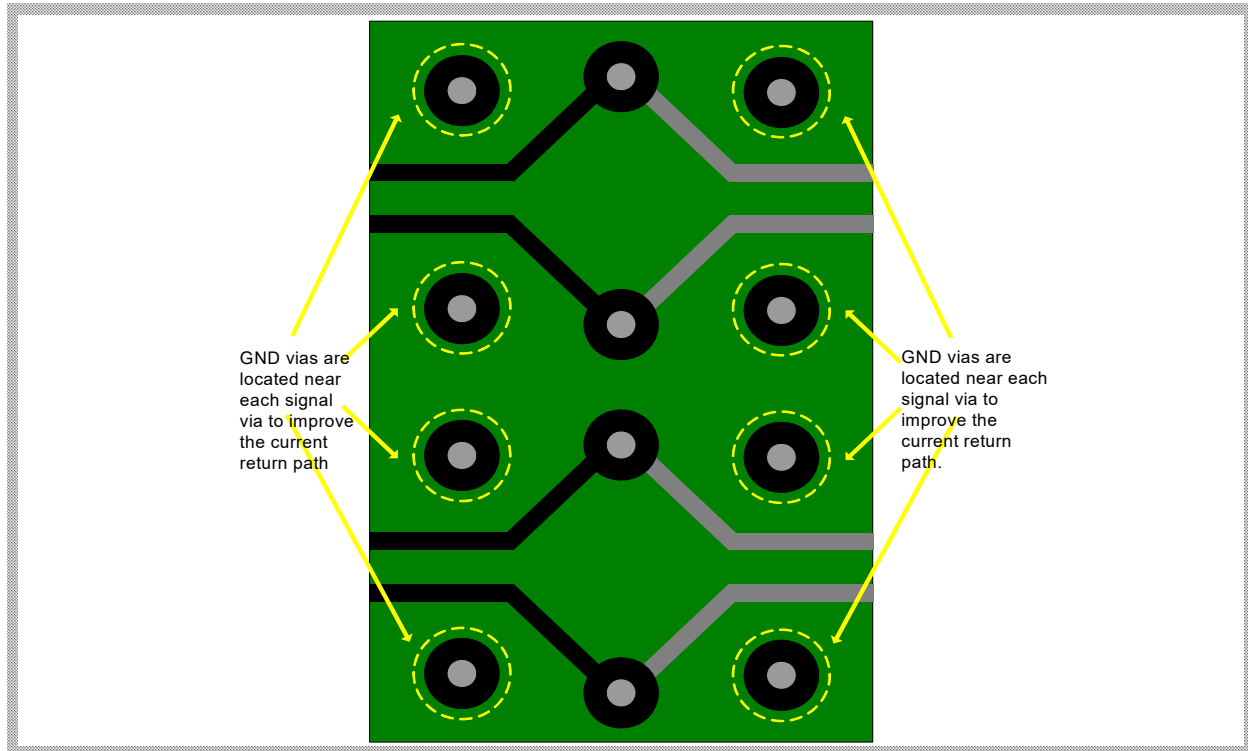


Figure 12-18. Return Path Vias for Differential Signals - Optimal Example

If the PCB fabrication process permits it, it's best to remove signal via pads on unconnected metal layers. See [Figure 12-19](#) and [Figure 12-20](#).

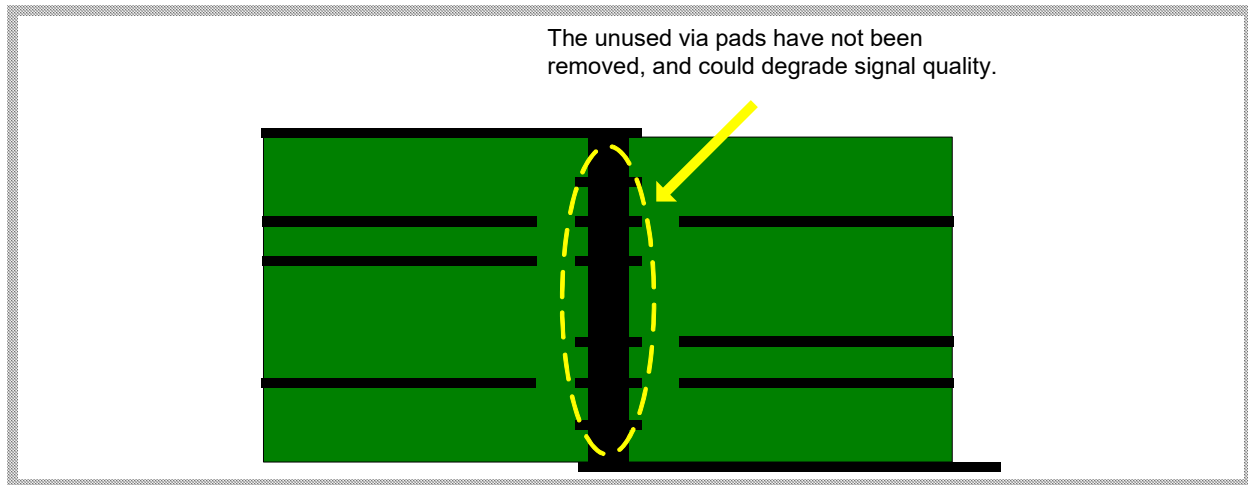


Figure 12-19. Signal Vias: Improper Padstack Example

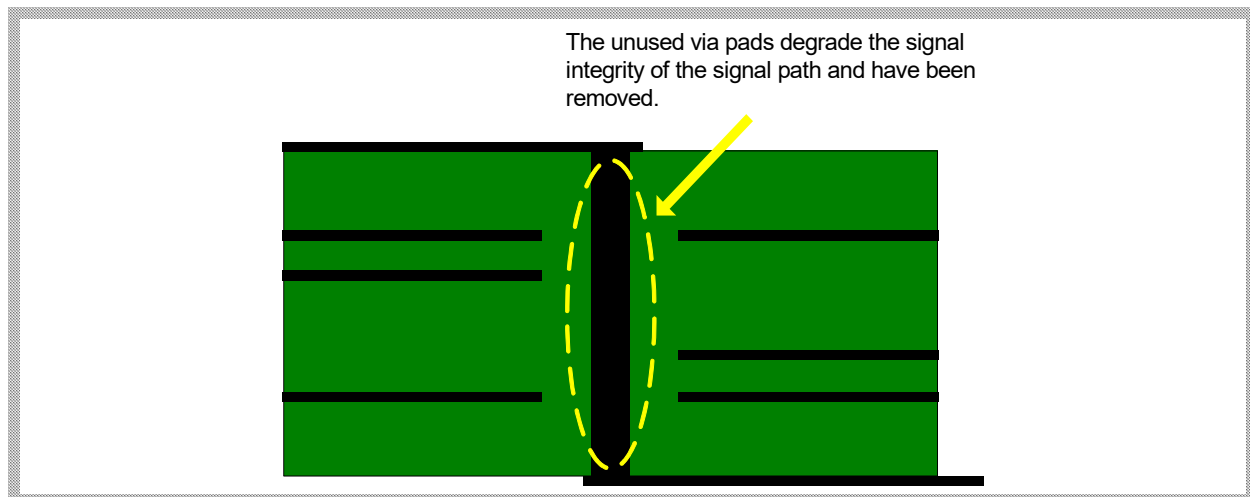


Figure 12-20. Signal Vias: Optimal Padstack

On metal layers where signal vias need to have via pads, it is desirable to reduce capacitance between the signal vias and ground-plane layers. The anti-pad diameters should be up to 20 mils larger than the via pad diameters. See [Figure 12-21](#). Clearance between the pad and the surrounding metal should be ≥ 10 mils.

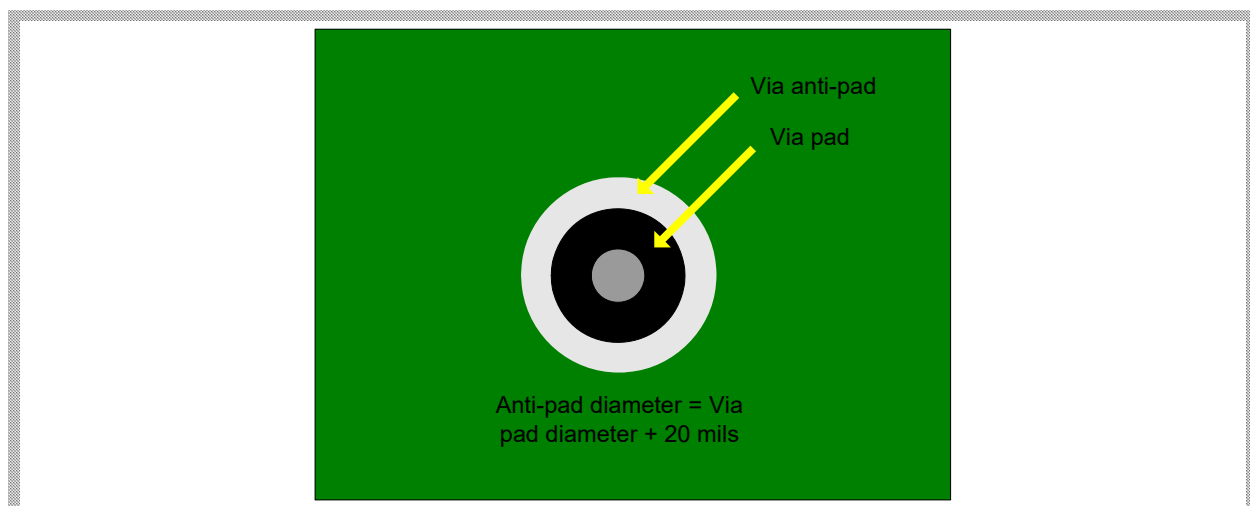


Figure 12-21. Anti-Pad Geometry

Each time differential signal vias pass through a plane layer, within each differential pair, the anti-pads should overlap. See [Figure 12-22](#), [Figure 12-23](#) and [Figure 12-24](#).

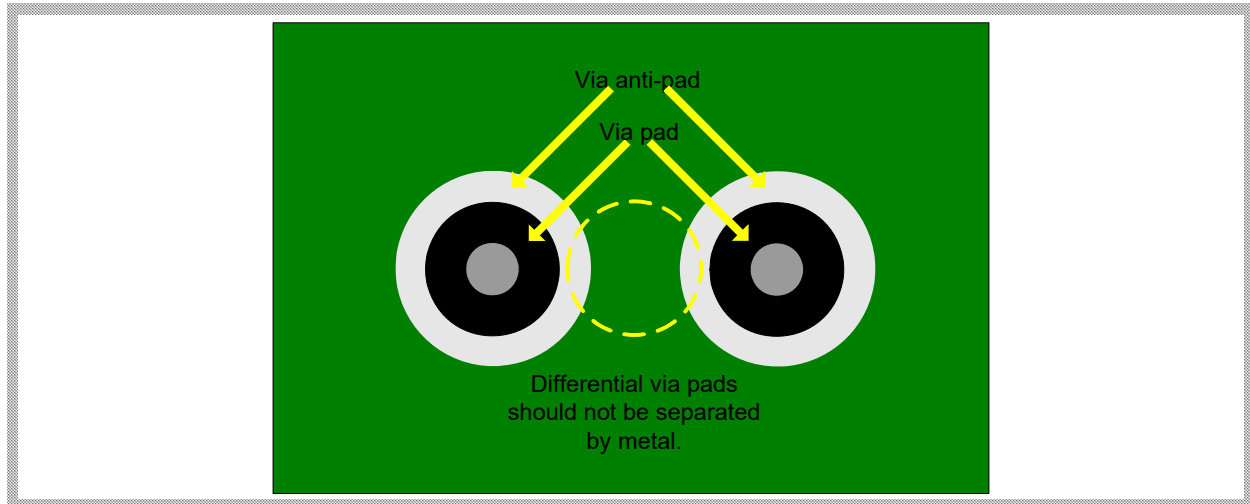


Figure 12-22. Differential Signal Vias: Improper Anti-pad Geometry

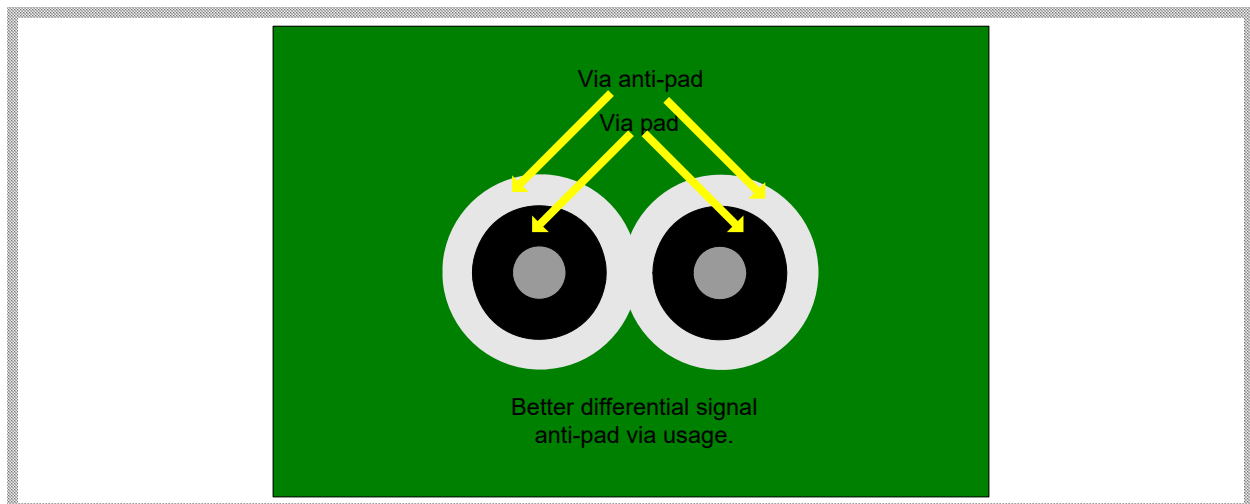


Figure 12-23. Differential Signal Vias: Acceptable Example

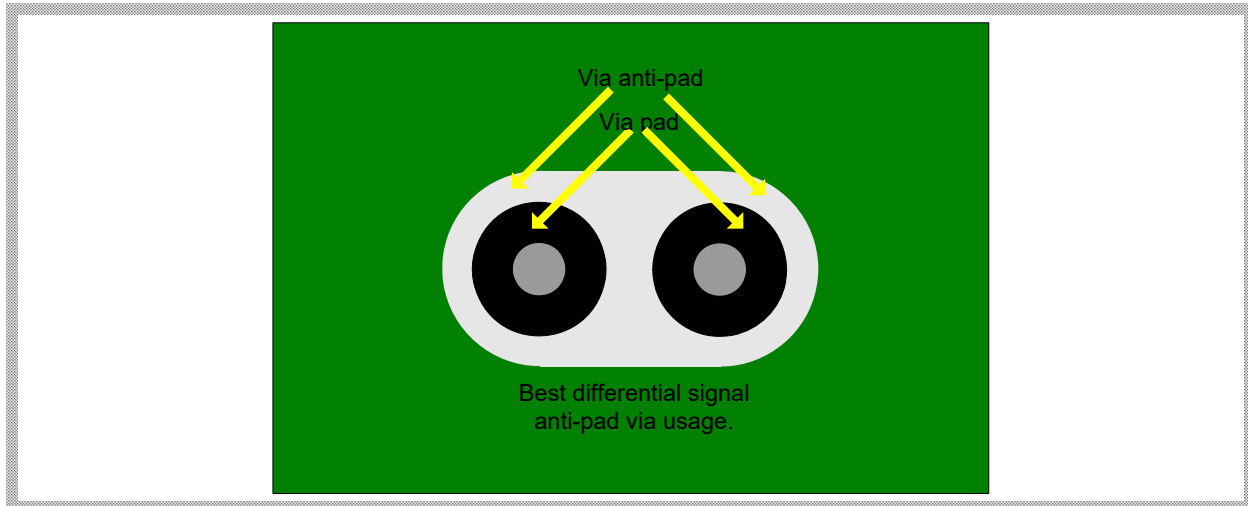


Figure 12-24. Differential Signal Vias: Optimal Example

12.12.9 Recommended Simulations

10GBASE-T signaling frequencies have frequency content in the range of 800 MHz and below so relatively short stubs, small discontinuities, and fairly small in-pair trace length differences can cause an undesirable increase in bit errors. Before ordering PCBs, verify that:

- Planned 10GBASE-T signal trace routing on the PCB complies with the interconnect characteristics recommended in IEEE 802.3an.
- With sufficient advance notice, Intel engineers can provide assistance under these conditions:
 - Trace routing should be optimized prior to the next steps – request a layout review (must be willing to provide board stack-up information and the MDI traces CAD artwork).
 - After MDI traces have been optimized, if the IEEE recommended electrical characteristics are still not being met, end-to-end MDI board channels S-parameter models should be extracted (preferably in Touchstone* S4p format) for additional investigative simulations by Intel signal integrity engineers. Please request the required S-parameter frequency range, step size, etc., before extracting Touchstone S-parameter models.

12.13 Bill Of Material (BOM)

Table 12-5 lists the BOM materials for all E610 LAN on Motherboard (LOM) designs.

Table 12-5. E610 NICs and LOMs

Component	Manufacturer	Manufacturer Part Number
25MHz	Hosonic TXC	E3SB25E001802E 7M25020018
156.25MHz	Diodes Incorporated NIHON DEMPA KOGYO	729-UX34F6203Z NP3225SC / NDKT02-00021-003 2/14
Flash	Adesto WinBond GigaDevice	AT25QF128A W25Q256JV, W25Q128JV GD258127D
Integrated Magnetic	Pulse	JT4-1114HL (1x1, PCIE NICs) JT11-1008HL (1x1, OCP3.0 NICs) JT6-1493NL (1x4, PCIE NICs) JT14-1013HL (1x4, OCP3.0 NICs)
	Belfuse	22550876 (1x1 PCIE NCIs) G09-43GC-012R (1x4 PCIE NICs) G52-4RUK-029A (1x4 OCP3.0 NICs)
	UDE	GD1-NT-0006 (1x1 PCIE NICs) GTK-NT-0001 (1x4 PCIE NICs)
RJ45	UDE	RT41-NT-00001 GSD-NT-001 (1x1 OCP3.0 NICs) GSD-NT-001 (1x4OCP3.0 NICs)
Discrete Magnetic	UDE	L22B005-0
Power Supplies FB	MURATA ELEC. NORTH AMERICA	BLM21PG220SN1D BLM21SP331SN1D BLM18AG601SN1 BLM15PX601SN1D BLM15BX182SN1D
	TDK CORPORATION OF AMERICA	MPZ2012S331AT000
	MAG. LAYERS SCIENTIFIC-TECHNICS CO., LTD	GMLB-201209-0022P-N8-EH
	TAIYO YUDEN, (USA), INC	LF BK1608HW601-TL
	SHENZHEN SUNLORD ELECTRONIC CO., LTD	GZ1608U601CTFP01
	TAI-TECH ADVANCE ELECTRONICS CO., LTD	HCB2012KF-220T60
	PULSE ELECTRONIC CORPORATION	BBSJ00160808601Y00

Chapter 13 Thermal Design Considerations

13.1 Introduction

This chapter can be used as an aid when designing a thermal solution for systems implementing the Intel® Ethernet Controller E610 (E610).

Properly designed thermal solutions provide adequate cooling to maintain the E610 product case temperature T_{case} (or junction) at or below thermal specifications. The E610 should function properly if case temperatures are kept at or below those presented. Ideally this is accomplished by providing a low local ambient temperature airflow and creating a minimal thermal resistance to that local ambient temperature.

By maintaining the case (or junction) temperature at or below the specified limits, a system designer can ensure the proper functionality, performance, and reliability of the E610. Operation outside the functional limits can cause data corruption or permanent damage to the E610.

The simplest and most cost-effective method to improve the inherent system cooling characteristics is through careful chassis design and placement of fans, vents, and ducts. When additional cooling is required, component thermal solutions can be implemented in conjunction with system thermal solutions. The size of the fan or heat sink can be varied to balance size and space constraints with acoustic noise.

13.2 Measuring the Thermal Conditions

This section provides a method for determining the operating temperature of the E610 in a specific system based on the junction and case temperature. Case temperature is a function of the local ambient and internal temperatures of the E610. The junction temperature can be read with the Intel software tools. This document specifies a maximum allowable T_{J-MAX} and T_{C-MAX} values for the E610. For conversion formula details, refer to [Section 13.6](#).

13.3 Thermal Considerations

In a system environment, the temperature of a component is a function of both the system and component thermal characteristics. System-level thermal constraints consist of the local ambient temperature at the component, the airflow over the component and surrounding board, and the physical constraints at, above, and surrounding the component that might limit the size of a thermal enhancement (heat sink).

The component's case/die temperature depends on:

- Component power dissipation
- Size
- Packaging materials (effective thermal conductivity)
- Type of interconnection to the substrate and motherboard
- Presence of a thermal cooling solution

- Power density of the substrate, nearby components, and motherboard

All these parameters are pushed by the continued trend of technology to increase performance levels (higher operating speeds, MHz) and power density (more transistors). As operating frequencies increase and packaging size decreases, the power density increases, and the thermal cooling solution space and airflow become more constrained. The result is an increased emphasis on system design to ensure that thermal design requirements are met for each component in the system.

13.4 Importance of Thermal Management

The thermal management objective is to ensure that all system component temperatures are maintained within functional limits. The functional temperature limit is the range in which the electrical circuits are expected to meet specified performance requirements. Operation outside the functional limit can degrade system performance, cause logic errors, or cause device and/or system damage.

Temperatures exceeding the maximum operating limits might result in irreversible changes in the device operating characteristics.

Note: The E610 has temperature sensors internal to the package, and automatically forces all Ethernet links down when the upper fatal temperature threshold (115 °C) is reached.

Also note that sustained operation at component maximum temperature limit might affect long-term device reliability.

13.5 Packing Terminology

Table 13-1. Packaging Terminology

Term	Description
Ambient	Refers to local ambient temperature of the bulk air approaching the component. It can be measured by placing a thermocouple approximately 1 inch upstream from the component edge.
BLT	Bond Line Thickness — Final settled thickness of the Thermal Interface Material (TIM) after installation of the heat sink.
CTE	Coefficient of Thermal Expansion — The relative rate a material expands during a thermal event.
FCBGA	Flip Chip Ball Grid Array package — A surface-mount package using a combination of flip chip and BGA structure whose PCB-interconnect method consists of Pb-free solder ball array on the interconnect side of the package. The die is flipped and connected to an organic build-up substrate with C4 bumps.
Junction	Refers to a P-N junction on the silicon. In this document, it is used as a temperature reference point (for example, Θ_{JB} refers to the "junction" to ambient thermal resistance).
LFM	Linear Feet per Minute — Airflow.
T_A	Local ambient temperature.
T_C	Case temperature — Temperature at geometric center of dies or over mold top surface.
T_j	Junction temperature — Maximum temperature of die active surface, i.e. hot spot.
TDP	Thermal Design Power — The estimated maximum possible/expected power generated in a component by a realistic application. Thermal solutions should be designed to dissipate this power level. TDP is not the peak power that the component can dissipate.
TIM	Thermal Interface Material — A conductive material used between the component and heat sink to improve thermal conduction.

Table 13-1. Packaging Terminology [continued]

Term	Description
Θ_{JB} (Theta JB)	Thermal resistance junction-to-board, °C/W.
Θ_{JC} (Theta JC)	Thermal resistance junction-to-case, °C/W.
Ψ_{JC} (Psi JC)	Junction-to-case (top of package) thermal characteristic parameter, defined by $(T_J - T_C) / TDP$. Ψ_{JC} does not represent thermal resistance, but instead is a characteristic parameter that can be used to convert between T_J and T_C when knowing the total TDP. This parameter can vary by environment conditions like heat sink and airflow.

13.6 Thermal Specifications

The following considerations should be made when designing a proper thermal solution for the E610:

- To ensure proper operation of the E610, the thermal solution must maintain a junction temperature at or below 105 °C for functionality and meet the TDP power specs. The device is tested to 105 °C.
- The E610 might experience temperature excursions to 125 °C for no more than 4% of the operating lifetime of the product. Functionality is not tested at this temperature, but there is no reliability impact if the device stays within these boundaries. Normal operation is expected to resume when the silicon returns to a max junction temperature of 105 °C.

System-level or component-level thermal enhancements are required to dissipate the generated heat to ensure the junction temperature never exceeds the maximum temperature. A good heat sink design and adequate system airflow are critical to dissipate the E610's high power.

To develop a reliable, cost-effective thermal solution, all the system variables must be considered:

- Use system-level thermal characteristics and simulations to account for individual component thermal requirements.
- A good thermal interface material (TIM) between the heat sink and IHS must be applied correctly.

The E610 compact thermal model is part of the Intel® Ethernet Controller E610 Thermal Mechanical Design Package (RDC ID [789527](#)) and can be downloaded from the [Intel Resource and Documentation Center](#). It is strongly encouraged that the E610 thermal model be used within a larger system model to properly design a system level thermal solution.

Keep the following in mind when reviewing the data that is included in this document:

- All data is preliminary and is not validated against physical samples.
- Your system design might be significantly different.
- A larger board with more layers might improve the E610 thermal performance.

The relationship between the parameters in [Table 13-2](#) are as follows:

$$\Psi_{JC} = (T_J - T_C) / TDP$$

$$T_{C-MAX} = T_{J-MAX} - (\Theta_{JC} * TDP)$$

Table 13-2. E610 Thermal Specifications

Parameter	E610		Notes
TJ-Max	105 °C		
TDP	Package Total	4.47 W	Max TDP; characterized at T _j = 105 °C. For a power down by each power rail and operational conditions, refer to Section 11.4, "Power Dissipation" .
	Controller Die	1.65 W	
	PHY1 Die	1.41.W	
	PHY2 Die	1.41.W	
θ _{JC}	10.83 °C /W		Thermal resistance junction-to-case.
θ _{JB}	6.98 °C /W		Thermal resistance junction-to-board.

13.7 Package Mechanical Attributes

The E610 comes in one package size:

- E610 — 14mm x 16 mm FCBGA with overmold.

Note: Be sure you use the official drawings, shown in [Section 11.7](#), for your detailed design. These drawings can be downloaded as part of the Intel® Ethernet Controller E610 Thermal Mechanical Design Package (RDC ID [789527](#)) from the [Intel Resource and Documentation Center](#).

For the maximum loads for heat sink to PCB attachment, refer to [Section 11.6.2, "Heat Sink Mechanical Load Limits"](#).

Chapter 14 Reliability, Diagnostics and Testability

14.1 Reliability

14.1.1 ECC Support and ECC Error Flow

Memories in the E610 are protected by ECC (ECC bits are added to the memory on write and compare on read). There are two types of ECC errors:

- Correctable ECC error — When the memory line has a single bit error, the ECC mechanism corrects it. This is done within the memory shell/wrapper and the flow continues as normal.
- Uncorrectable ECC error — When the memory line read has more than a single ECC error, it cannot be recovered by the ECC mechanism. The text that follows describes how the E610 reacts to such an event.

Uncorrectable errors are handled as follows:

- Device blocks data from going to an external link and blocks any new PCIe master transaction, but the erroneous transaction may reach host memory.
- Recovery then depends on the memory where the error happens:
 - An error takes place in the core or PCIe domains.
 - An ECC error check in the shadow RAM is enabled via the Shadow RAM ECC Enable bit in the NVM.
 - The E610 reloads the shadow RAM from the NVM.
 - If the *ECC_FWRST_EN* bit is set to 1b, the E610 generates a firmware reset (which in turn reloads the relevant sections into hardware).
 - An error takes place in the PCIe domain.
 - If an error is in data to be sent out, the TLP is sent with an EDB. Else, the link goes to a link-down state.
 - If an error is in data to be sent out, the TLP is sent with an EDB. Else, the link goes to a link-down state.

14.2 Link Loopback Operations

Loopback operations are supported by the E610 to assist with system and device debug. Loopback operation can be used to test transmit and receive aspects of software device drivers, as well as to verify electrical integrity of the connections between the E610 and the system (such as PCIe bus connections, etc.).

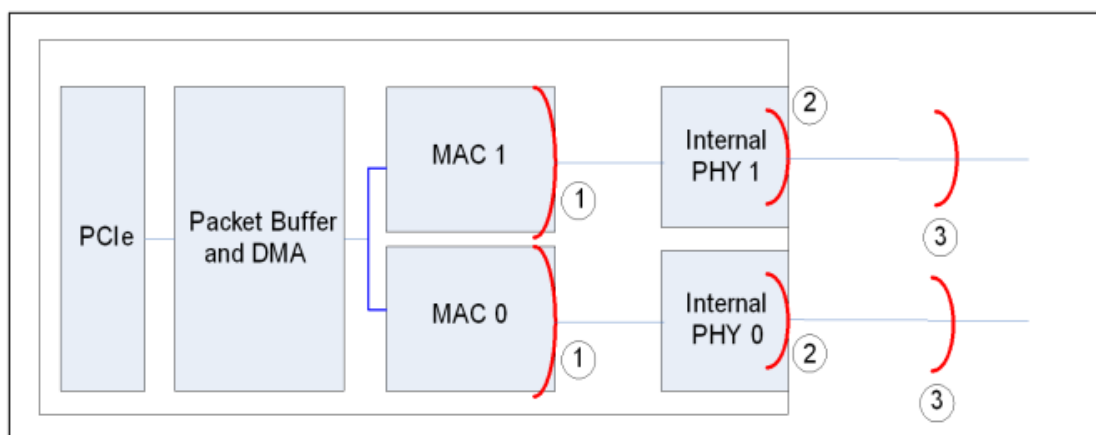


Figure 14-1. Loopback Modes

14.2.1 Firmware Logging Overview

Firmware logging is a set of commands to retrieve logging information gathered by firmware during its normal operation. The gathering of the information is based on different modules and severity.

The available modules are described below divided by information types. The software controlling the logs may expose an API that allows the use to gather information per information type. For example by using a command of the form:

```
fw_log_severity_config --option=NVM -log_level=3
```

The software can then instruct the firmware to change the logging settings using the Set Firmware Logging Configuration (0xFF30) CSR based AQ command.

Table 14-1. Information Types

Information Types	Module Name	Module #
Initialization	CTRL	0x1
NVM	NVM	0xE
	Authentication	0xF
	VPD	0x10
IO	I2C	0x5
	SDP	0x6
	MDIO	0x7

Table 14-1. Information Types [continued]

Information Types	Module Name	Module #
Link Management	Link Management	0x2
	Link Topology Detection	0x3
	Dreadnought Lake	0x4
AQ Interface	Admin Queue	0x8
Manageability	Manageability	0x1A
Infrastructure	Watchdog	0x18
	Task Dispatcher	0x19
	Generic	0x0
	IOSF	0x11
	SyncE	0x1B
	Time Sync	0x1D
	Health	0x1C
	PF Registration	0x1E
	Module Version	0x1F
QoS	DCB	0xC

The following severity levels are defined:

1. No Log
2. Error
3. Warning
4. Normal
5. Verbose

The default types of information gathered is defined in the NVM in the Firmware Logging Defaults Section in the PFA. The default values are described in [Section 14.2.1.1](#).

14.2.1.1 NVM Default Firmware Log Configuration

The NVM enables the default configuration as described in Table 14-2. The software base drivers provide the ability to override these defaults: ON=1, OFF=0.

Table 14-2 describes NVM default configuration expected values:

Table 14-2. NVM Default Firmware Log Configuration

Module #	NVM Word Numbering	Module Name	Error	Warning	Normal	Verbose
0x0	N=0	General	ON	ON	OFF	OFF
0x1		Control (Resets + Autoload)	ON	ON	ON	OFF
0x2		Link Management	ON	OFF	ON	OFF
0x3		Link Topology Detection	ON	OFF	OFF	OFF
0x4	N=1	Dreadnought Lake	ON	OFF	OFF	OFF
0x5		I2C	OFF	OFF	OFF	OFF
0x6		SDP	OFF	OFF	OFF	OFF
0x7		MDIO	OFF	OFF	OFF	OFF
0x8	N=2	Admin Queue	ON	OFF	OFF	OFF
0x9		N/A (Not applicable)	OFF	OFF	OFF	OFF
0xA		N/A	ON	OFF	OFF	OFF
0xB		N/A	ON	OFF	OFF	OFF
0xC	N=3	DCB	ON	OFF	OFF	OFF
0xD		NetProxy	ON	ON	ON	OFF
0xE		NVM	ON	OFF	OFF	OFF
0xF		Authentication	ON	OFF	OFF	OFF
0x10	N=4	VPD	ON	OFF	OFF	OFF
0x11		IOSF	ON	OFF	OFF	OFF
0x12		N/A	ON	OFF	OFF	OFF
0x13		N/A	ON	OFF	OFF	OFF
0x14	N=5	N/A	ON	OFF	OFF	OFF
0x15		N/A	ON	OFF	OFF	OFF
0x16		N/A	ON	OFF	OFF	OFF
0x17		N/A	ON	OFF	OFF	OFF
0x18	N=6	Watchdog	ON	OFF	OFF	OFF
0x19		Task Dispatcher	ON	OFF	OFF	OFF
0x1A		Manageability	ON	OFF	OFF	OFF
0x1B		SyncE	ON	OFF	OFF	OFF
0x1C	N=7	Health	ON	OFF	OFF	OFF
0x1D		Time Sync	ON	OFF	OFF	OFF
0x1E		PF Registration	ON	OFF	OFF	OFF
0x1F		Module Version	ON	OFF	OFF	OFF

14.2.1.2 Persistent Crash Log

To provide ability to diagnose firmware crash situations, when firmware is reset as a result of Watchdog expiration or other unrecoverable error, it provides pre-EMPR log information that was stored. Pre-EMPR log information is provided to the software driver on the next boot. To support that, firmware copies at least 256 bytes of latest logs to EMPR persistent memory (a memory that is not cleared during EMPR) inside watchdog exception handler, just before EMPR is triggered.

14.2.2 Health Status Commands

Table 14-3 list the available Health Status commands.

Table 14-3. Health Status Admin Commands

Command Name	Opcode	Type	Reference
Set Health Status Configuration	0xFF20	Direct	Section 14.2.2.1
Get Supported Health Status Codes	0xFF21	Direct	Section 14.2.2.2
Get Health Status Event	0xFF22	Direct	Section 14.2.2.3
Clear Health Status	0xFF23	Direct	Section 14.2.2.4

14.2.2.1 Set Health Status Configuration (0xFF20)

For user-visible events (like “Link configuration mismatch detected”) offering actionable feedback, firmware sends events to PFs subscribed to Health Status messages. These messages come in two varieties:

- PF Health Status
- Global Health Status

PF Health Status events are specific only to one PF and not to the global device configuration. For example, a specific port issue is only reported to the PF connected to that port, and only if that PF has enabled PF Health Status. Firmware does a Port-to-PF lookup to pick the appropriate PF. If, for any reason, the Port-to-PF lookup fails, firmware issues the event as a Global Health Status. Software can optionally listen to PF Health Status across all PFs by setting the Enable All PF Health.

Global Health Status are events that affect the global device configuration. For example, a specific data path issue shall be reported to all PFs that have enabled Global Health Status.

The Health Status Configuration for a PF always resets to “Disabled” for both PF Health Status and Global Health Status during PFR and other High Resets, while existing error statuses remain in firmware memory and are not cleared.

Table 14-4. Set Health Status Configuration Command and Response

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 10.11.2.4.1.1 and Section 10.11.2.4.1.2 for details.
Opcode	2-3	0xFF20	Command opcode.
Datalen	4-5	0x0	
Return Value	6-7		Return value. Zeroed by driver. Written by firmware. Firmware returns EPERM when Tools Q enables PF Health Status.
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
Enable PF Health Status	16.0		If this bit is set, PF-specific Health Status is enabled for this PF.
Enable All PF Health Status	16.1		If this bit is set, all PF-specific Health Status is enabled for this PF.
Enable Global Health Status	16.2		If this bit is set, Global Health Status is enabled for this PF.
Reserved	16.3-31		Reserved. Ignored by firmware.

Following the PF's registration, firmware pushes existing Device Health Data to the registering PF through an AQ event.

14.2.2.2 Get Supported Health Status Codes (0xFF21)

Software can request a list of all supported Health Status Codes by issuing this command.

Table 14-5. Get Supported Health Status Log Events Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 10.11.2.4.1.1 for details.
Opcode	2-3	0xFF21	Command opcode.
Datalen	4-5	0x0	
Return Value	6-7		Return value. Zeroed by driver. Written by firmware.
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16-23		Reserved. Ignored by firmware.
Data Address High	24-27	Buffer Address	Address of response buffer.
Data Address Low	28-31	Buffer Address	

Table 14-6. Get Supported Health Status Log Events Response

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 10.11.2.4.1.2 for details.
Opcode	2-3	0xFF21	Command opcode.
Datalen	4-5	0x0	
Return Value	6-7		Status of request. SUCCESS - Firmware has provided the supported Health Status Codes. ENOMEM - Buffer provided by software was too small. Software can use the <i>Health Status Code Count</i> field to calculate the necessary buffer to call this command again.
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
Health Status Code Count	16-17		Written by firmware to indicate the number of Health Status Codes supported by firmware.
Reserved	18-23		Reserved. Ignored by firmware.
Data Address High	24-27	Buffer Address	Address of response buffer.
Data Address Low	28-31	Buffer Address	

The response buffer for this command contains a *Health Status Code Count* number of Health Status Codes.

Table 14-7. Set Firmware Configuration Command Buffer

Bytes	Remarks
0-1	Health Status Code

14.2.2.3 Get Health Status Event (0xFF22)

This Admin Queue event provides the Health Status information for both PF Health Status and Global Health Status.

Software subscribed to these events is expected to translate the Health Status Code and additional parameters (as required) into user-visible information. If software does not know how to interpret a Health Status Code, it can inform the user or silently discard the message. The structure below describes both AQ Command and Event structure, which has is same format.

Table 14-8. Get Health Status Event/Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 10.11.2.4.1.1 and Section 10.11.2.4.1.2 for details.
Opcode	2-3	0xFF22	Command opcode.
Datalen	4-5		Number of bytes delivered in this event, if any.
Return Value	6-7	0x0	
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
Health Status Code Count	16-17		Command: Reserved Event/Response: Number of Health Status Code provided in Buffer.
Reserved	18-23		Reserved. Ignored by firmware.
Data Address High	24-27	Buffer Address	Address of response buffer.
Data Address Low	28-31	Buffer Address	

“Internal Data” fields from AQ command are defined per Health Status Code (similar to Device Capabilities AQ command).

Table 14-9. Get Health Status Event Response Buffer

Bytes	Value	Remarks
0-1	Health Status Code	Health Status Code that resulted in the generation of this event.
2-3	Event Source	Indicator as to the potential source of this event. 1 = For PF Health Status. PF ID shall be set to the PF number of the PF associated with this event in upcoming parameters. 2 = For Port Health status. 3 = For Global Health Status. All other values are reserved.
4-7	Internal Data 1	This code represents additional data associated with the source of this event. When this field is set to 0xDEADBEEF, no Internal Data Code has been provided.
8-11	Internal Data 2	

In case of multiple statuses at the same time, smaller index in buffer status array - older status.

Following the AQ command, existing all PFs, current PFs, and Global Errors are reported according to PF configured/registered policy. For tools AQ, only all PFs Statuses and Global are reported. Each reported error is cleared in firmware.

The Internal Data Code is optional and can be omitted from any and all user-facing status reports. The Internal Data Code can be anything. It is intended to be used to aide in differentiating between identical events generated at differing positions in code. For example, this can be a line number or internal variable information. It is not intended to be used for parameters. All parameters associated with a *Health Status Code* must be provided in the response buffer.

14.2.2.4 Clear Health Status Command (0xFF23)

This Admin Queue Command Clears firmware internally stored Health Status.

Table 14-10. Clear Health Status Event/Command

Name	Byte.Bit	Value	Remarks
Flags	0-1		See Section 10.11.2.4.1.1 and Section 10.11.2.4.1.2 for details.
Opcode	2-3	0xFF23	Command opcode.
Datalen	4-5		Number of bytes delivered in this event, if any.
Return Value	6-7	0x0	
Cookie High	8-11	Cookie	Opaque value copied by firmware into the completion of this command.
Cookie Low	12-15	Cookie	Opaque value copied by firmware into the completion of this command.
Reserved	16-31		Reserved.

14.2.2.5 Health Status Codes

Table 14-11. Health Status Codes

Info	Description	AQ Command: Health Status Code	AQ Command: Additional Data 1	AQ Command: Additional Data 2
Recovery Mode	Firmware is in recovery mode.	0x500	FWSM.EXT_ERR_IND - Failure module information	
Reset Counters	Firmware experience resets in loop.	0x1000	GLGEN_RSTAT information for EMPR/CORER/GLOBR. EMPR Reason (WD, Exception, ECC etc.)	
PFR failure	Firmware failed to execute PF reset because of failure in module X.	0x1001	PF_ID	
Firmware boot problem	Firmware load did not succeed,	N/A	FWSM - boot stages	
PCIe status	PCIe IF problem.	N/A	Open - Need to define info	
Flash access failure	Firmware cannot access flash. Possible broken chip.	0x501	Access Type 0- Read 1- Write 2 - Erase	
NVM image broken	NVM image broken. Authentication failed.	0x502	None	
OROM image broken	Authentication failed.	0x503	None	
Package image broken	Authentication failed.	0x504	None	
NVM image incompatible	Incompatible device or module ID.	0x505	Expected device ID	Expected module ID
OROM incompatible	Incompatible device or module ID.	0x506	Expected device ID	Expected module ID
NVM Security version violation	Non-permitted Security version downgrade attempt.	0x507	Minimal Security version expected	

Table 14-11. Health Status Codes [continued]

Info	Description	AQ Command: Health Status Code	AQ Command: Additional Data 1	AQ Command: Additional Data 2
OROM Security version violation	Non-permitted Security version downgrade attempt.	0x508	Minimal Security version expected	
DCBx Reverted to default	DCBx MIBs are reverted to default, following bad MIB received.	0x509	Port #	MIB ID
Last failed AQ command since last software API call	Tells us which software driver API was invoked during the last AQ command failure and tells us which AQ command failed in the same flow, if any.	0x1002	Failed AQ Opcode	PF_ID
MNG timeout	TBD	0x50A		
Reset command received from BMC	Reset Was triggered by Manageability.	0x50B	Reset Type: TBD	
Last failed MNG command	Manageability command failed.	0x50C	MNG type	Failed Opcode
Resource allocation failure	Information about last case firmware reject PF driver resource request.	0x50D	Resource type	PF_ID
Module Error	Unspecified/Unknown Module, strict mode.	0x101	Port #	
	Module type not supported.	0x102	Port #	
	Module not qualified.	0x103	Port #	
	Module communication error.	0x104	Port #	
	Unresolved module conflict.	0x105	Port #	
	Module not present.	0x106	Port #	
Module Info	Underutilized module.	0x107	Port #	
	Unspecified/Unknown Module, lenient mode.	0x108	Port #	
	Module diagnostic feature not implemented.	0x109	Port #	
Configuration Error	Unsupported loopback.	0x10A	Port #	
	Invalid Link Configuration.	0x10B	Port #	
	Temporary hardware access error.	0x10C	Port #	
	Unreachable port.	0x10D	Port #	
Configuration Info	Port Speed limited due to module.	0x10F	Port #	
	Parallel Fault detected.	0x110	Port #	
	Underutilized port.	0x111	Port #	
Fatal Error	LOM Topology Netlist corrupted.	0x112		
	Unrecoverable netlist error.	0x113	Port #	
	Topology conflict.	0x114	Port #	
	Hardware access error.	0x115	Port #	
	Unrecoverable runtime error.	0x116	Port #	
	DNL initialization failure.	0x117		
Netlist image broken	Authentication failed.	0x510	None	

Table 14-11. Health Status Codes [continued]

Info	Description	AQ Command: Health Status Code	AQ Command: Additional Data 1	AQ Command: Additional Data 2
Netlist image incompatible	Unsigned netlist update when netlist signing is opted in.	0x511	None	
Netlist Security version violation	Non-permitted Security version downgrade attempt.	0x512	Minimal Security version expected	

14.3 Firmware Recovery Mode

14.3.1 Overview

Recovery mode is intended to recover from a fatal failure scenario in which the device is not accessible to the host, meaning firmware is non-responsive.

A PDoS scenario might occur by a misconfiguration done by a malicious software that corrupts device-specific information that is vital for firmware operation and/or its enumeration over the PCIe.

The purpose of the Firmware Recovery mode is to enable the software tools to update or rollback the firmware and/or device configuration so that the fatal error is resolved.

14.3.1.1 Supported Failure Scenarios by Firmware

Possible supported failure scenarios that are covered by firmware:

- Firmware boot failure (for example, fails to authenticate NVM bank).
- Firmware initialization failure on POR/EMPR (firmware reset).
- Master reset / MAC reset firmware flow failure.
- Repeated Watchdog reset.
- Repeated Memory exception.

14.3.1.2 Dependencies for Recovery Firmware

Firmware does not run Recovery mode in cases where the error is in one of following firmware components:

- ROM firmware.
- Mini-loader firmware (boot loader).
- Recovery Firmware section corrupted or erroneous.

In case any of the above scenarios occur, the device enters Blank Flash Mode (BFP) in which security is off and the entire NVM can be directly accessed by software via CSRs.

14.3.2 Recovery Flows

When firmware detects a failure scenario, it first attempts to automatically rollback to the non-active NVM bank (see [Section 14.3.2.1](#)).

If rollback fails or the non-active bank is not valid, firmware attempts recovery (see [Section 14.3.2.2](#)).

Both flows are run automatically by firmware. Software has no ability to trigger these flows.

Exit from a recovery flow to normal operation mode can be done following successful NVM update flow triggered by the software tool or POR. In normal operation mode, device runs the firmware from the valid NVM bank.

In case of exception or Watchdog timeout, each recovery flow is attempted up to a pre-defined amount of times (3) before firmware decides to try the next recovery flow. Before every attempt, firmware restarts the EMP firmware by issuing a graceful EMPR. Firmware restarts the counting in case firmware is running successfully for a pre-defined (hard-coded) time (five hours).

14.3.2.1 Automatic Rollback

Firmware automatically tries to rollback to “old” operational firmware located in the invalid NVM bank in cases where operational firmware from the valid NVM bank repeatedly enters a PDoS scenario. The invalid NVM bank must pass authentication and other security checks, such as SREV downgrade protection.

After rollback is successful, the banks are swapped, and the “rollback” bank becomes the active bank for the next reset.

14.3.2.2 Recovery Firmware

Recovery Firmware is a standalone bank that contains basic functionality to allow software tools to update firmware/NVM and/or device configuration.

The Recovery Firmware section is separately signed and is upgradeable as part of full NVM bank.

Recovery Firmware is invoked automatically by mini-loader firmware in cases where Rollback firmware repeatedly enters a failure scenario.

Recovery Firmware is run from the latest active bank disregarding rollback attempt.

14.3.2.2.1 Recovery Firmware Supported Features

14.3.2.2.1.1 Recovery Mode Admin Commands

Recovery Firmware supports a reduced set of admin commands, as listed below. Other commands are rejected with BAD_OPCODE error. Admin commands are only supported from PF0 and the “CSR-Based Firmware Admin Queue for Tools” (see [Section 10.11.2](#)).

- [Section 10.11.2.7.1, “Get Version \(0x0001\)”](#)
- [Section 10.11.2.7.3, “Set PF Context \(0x0004\)”](#)
- [Section 10.11.2.7.5, “Release Resource Ownership \(0x0009\)”](#)
- [Section 3.3.10.1, “NVM Read \(0x0701\)”](#)
- [Section 3.3.10.2, “NVM Erase \(0x0702\)”](#)

- [Section 3.3.10.3, "NVM Write \(0x0703\)"](#)
- [Section 3.3.10.8, "NVM Write Activate \(0x0707\)"](#)

14.3.2.2.1.2 Recovery Mode Admin Command Usage Guidelines

Usage guidelines are as follows:

- The NVM Update and NVM Erase commands support only NVM bank and extended TLV bank.
- The NVM Activate command does not support selected preservation (01) of the PFA.
- The NVM Read command supports NVM bank, Extended TLV bank, and read of the entire flash in flat mode (0x0).
- The NVM update tools can read the EETRACK ID using the NVM Read AQC (0x701). The EETRACK ID is not available in Shadow RAM during recovery.
- Resource request and release commands support only NVM resource (0x1).
- Set PF context supports only PF0.

14.3.2.2.2 Reset Flows

14.3.2.2.2.1 Port (Master) reset / MAC reset

Recovery Firmware will perform Port / MAC reset with no autoloading or initializations.

Firmware will acknowledge to software that reset is done and AL is done as it does in operational firmware.

14.3.2.2.2.2 PFR

Recovery Firmware does not perform any device initialization related to data path, and only acknowledges to software that the reset is done, as it does in operational firmware.

14.3.2.2.2.3 PCIR/PERST

Recovery Firmware will perform minimal configuration to enable AQ communications.

Recovery Firmware will autoloading PCIe registers from the PFA inside the 'Factory Settings' section instead of active PFA in the SR bank and from the signed EML section.

Firmware will acknowledge to software that reset is done and AL is done as it does in operational firmware.

PCIR registers AL is checked against a white-list that is loaded from the signed EML section.

14.3.2.2.2.4 POR/EMPR

Recovery Firmware will autoloading POR registers from PFA inside "Factory Settings" section of active PFA in the SR bank and from the signed EML section.

POR registers AL is checked against a white-list that is loaded from the signed EML section.

14.3.2.2.3 Unsupported Features

Recovery firmware does not support:

- Link
- Traffic
- Manageability (BMC)

Recovery firmware only supports firmware update flows from host.

14.3.3 Operation Mode Software Identification

Firmware indicates operation mode via the *FW_MODES* field (Bits 2:0) in the GL_MNG_FWSM CSR (see [Section 8.2.2.24.10](#)). Bits are exclusive.

- Bit 0 = Debug mode indication (0:non-debug mode, 1:debug mode)
- Bit 1 = Recovery mode indication (0:normal mode, 1:recovery mode)
- Bit 2 = Rollback indication (0:normal mode, 1:rollback mode)
- Rollback and recovery indications are valid until next firmware update or next POR. Following firmware update or POR, they are cleared.

Chapter 15 Glossary and Acronyms

This section defines terms and acronyms commonly used throughout this specification. Another source for this type information is [Section 1.6](#) which contains some of the acronyms defined in the industry standards (e.g. MII, MPA, Verbs, NC-SI, etc).

Table 15-1. Glossary and Acronyms

Acronym/Term	Description
AN	Auto-negotiation — An IEEE Ethernet method for exchanging PHY layer parameters such as speed and duplex mode between link partners.
BAR	Base Address Register — Standard registers defined in PCI Config space that define how an I/O adapter responds to host memory-mapped I/O requests.
BER	Bit Error Rate — Number of bits received in error, divided by the total number of bits received.
BFP	Blank Flash Programming
BMC	Baseboard Management Controller — A BMC is a specialized embedded processor typically integrated on a server motherboard that reports the state of the server to the system administrator independent of the state of the server OS.
CEQ	Completion Event Queue — The E610 writes Completion events to this shared queue to make it easy for software to determine which CQ has new CQEs.
CQ	Completion Queue — A sharable queue which can contain one or more Completion Queue Entries. A Completion Queue is used to create a single point of completion notification for multiple Work Queues. The Work Queues associated with a Completion Queue may be from different QPs and of differing queue types (SQs or RQs).
CQE	Completion Queue Entry — Info that the E610 writes onto a Completion Queue during the process of performing a Completion.
Completion, Completed, Completes, etc.	When an RNIC has performed all functions specified for a given WQ operation, including Placement and Delivery, that WQ operation is said to be "completed". This can be determined by the Consumer through a Work Completion for Signaled Work Requests.
Consumer	A software process that communicates with the E610. The Consumer typically consists of an application program or an OS adaptation layer which provides some OS-specific API.
CSR	Control/Status Register — I/O adapter registers typically accessed by the host using memory-mapped I/O requests.
CRC	Cyclic Redundancy Check — An error detecting code commonly used to protect network transmissions.
DSI	Dataplane Switch-mode Interface — A software API for dataplane applications to allow direct attach of multiple users to device resources.
DTMF	Distributed Management Task Force
ECC	Error Correcting Code — A code commonly used to protect network transmissions which allows both detection and recovery from errors.
EMP	Embedded Management Processor — An embedded processor that handles device initialization and also device configuration based on commands received from an Admin Queue, a management interface (e.g. NC-SI), or from a network port.
EMPR	Embedded Management Processor Reset
FIPS	Federal Information Processing Standard — Published by NIST.
FLR	Function-level Reset — A capability defined in the <i>PCI Express Base Specification</i> that enables software to quiesce and reset a particular PCI Function in a MFD, without affecting the other PCI functions therein.
FW	Firmware — An executable program generally located in ROM or NVM and executed by an embedded processor.
HPC	High Performance Computing — A common term for the compute market of high performance applications
IFP	In-Field Programmable

Table 15-1. Glossary and Acronyms [continued]

Acronym/Term	Description
IOVM	I/O Virtual Machine - A special VM that manages the software switch and owns the physical NIC. It is responsible for the configuration and managing of the physical device.
IPG	Inter Packet Gap — A set of idle signals sent over the network to separate packets.
ITR	Interrupt Throttling — A method of interrupt moderation that guarantees a minimum gap between two consecutive interrupts.
KVM	Keyboard, Video, Mouse — Standard bundle of computer user I/O devices.
LACP	Link Aggregation Control Protocol — The IEEE protocol that enables the formation of Link Aggregation Groups (aggregation of one or more Ethernet links into a single logical link).
LLDP	Link Layer Discovery Protocol — The IEEE protocol that enables a server to advertise its identity, capabilities, and interconnections to other entities on an Ethernet fabric.
LLP	Lower Layer Protocol — The protocol layer beneath the protocol layer currently being referenced. For example, for TCP the LLP is IP.
LOM	LAN On Motherboard — When the E610 is integrated on a server motherboard, it qualifies as a LOM NIC.
MDIO	Management Data Input/Output Interface — A standard interface to connect a MAC to a PHY, used to access PHY registers.
MFD	Multi-Function Device — A PCI Device with more than one PCI function.
MFP	Multi-Function per Port — A PCI Device is classified as an MFP device when it can support more than one PCI Physical Function bound to a single network port. See also <i>Single-Function per Port (SFP)</i> .
ML	Mini-Loader — EMP firmware executed out of EMP internal SRAM and responsible for bringing up PCIe interface within PCIe specification's 100 ms time limit, then authenticating and launching the main EMP NVM firmware.
MSS	Maximum Segment Size — The largest amount of data that a network device can handle in a single, unfragmented piece.
MW	Memory Window — A subset of a Memory Region, which can be remotely accessed in a logically contiguous fashion. A Memory Window is identified by an STag, a Base TO, and a length, but also references an underlying Memory Region and has Access Rights.
NAC	Network Acceleration Complex — An IP that combines a E610 IP, a Highland Park IP, a CPM IP, and PHYs into a single networking IP.
Nearest Bridge Access	A multicast MAC Address used for DCBx exchange.
NIC	Network Interface Controller — An I/O adapter that enables a computer to communicate over a network.
NIST	National Institute of Standards and Technology — United States government commerce agency which defines security standards.
NSA	National Security Agency
NVM	Non-Volatile Memory
OEM	Original Equipment Manufacturer
OTP	One-Time Programmable
Packet	A unit composed of headers, data and footers that are sent or received by a device. Also known as a frame.
Page List	A list of physical addresses describing a set of memory pages, which specifies the page size, list of physical addresses, and offset to the start of the memory region within the first page. The starting physical addresses of each page is aligned on power-of-two addresses and the size of the page is a power of two. Note that it is possible for the starting offset to be an offset into the first page and to be of a byte granularity and the entire list may have an arbitrary length.
PF	Physical Function — A PCI Function that supports the PCI-SIG <i>Single Root I/O Virtualization and Sharing Specification</i> .
PFA	Preserved Field Area — NVM section used to store run-time configuration data that is not overwritten when the EMP firmware image is updated.
PLDM	Platform Level Data Model

Table 15-1. Glossary and Acronyms [continued]

Acronym/Term	Description
Quad	In this document, the term “quad” is defined as the set containing {source IP Address, dest IP Address, source port, dest port} taken from a TCP/IP packet.
Quanta	A configurable amount of bytes consisting of one or more packets.
RAD	Required Authorization for Debug
RNG	Random Number Generator
ROM	Read-Only Memory
RQ	Receive Queue — One of the two Work Queues associated with a Queue Pair. The Receive Queue contains Work Queue Elements that describe the Buffers into which data from incoming Send Operation Types is placed.
RSA	Rivest Shamir Adleman — Asymmetric cryptographic algorithm. The acronym comes from the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, who publicly described the algorithm in 1977.
RSS	Receive Side Scaling — A feature of Microsoft Windows OS that distributes receive packet processing to the different processors in a multi-processor system. Received packets are classified by the E610 under OS control into groups of conversations. Each group of conversations is assigned its own receive queue and receiving processor.
Scheduler Quanta	A “quanta” is a configurable amount of bytes which may consist of one or more packets. The Scheduler schedules a “quanta” in each scheduling decision.
SDPs	Software Definable Pins — The E610 device pins that have a high degree of software configurability and programmability. Also called “general purpose I/Os” (GPIOs).
SFP	Single-Function per Port — A PCI Device is classified as an SFP device when it can support only one PCI Physical Function bound to a single network port. See also <i>Multi-Function per Port (MFP)</i> . or Small Form-factor Pluggable — A small form-factor pluggable (SFP or SFP+) module is a compact, hot-pluggable transceiver that interfaces a network device like the E610 to a fiber optic or copper networking cable.
SHA	Secure Hash Algorithm
TAP	Test Access Port
TLV	Type, Length, Value — A technique used to encode messages in a data communication protocol. Each message is comprised of three sequential fields: A <i>Type</i> field, fixed in size and typically ~1 byte, which identifies the specific type of message and the format of information in the <i>Value</i> field. A <i>Length</i> field, fixed in size and typically ~1 byte, which defines the length of the <i>Value</i> field in octets. A <i>Value</i> field, variable in size, which contains all of the data specific to this message type. A protocol relevant to the E610 that uses TLV encoding is IEEE Link Layer Discovery Protocol (LLDP).
TSO	Transmit Segmentation Offload — Also known as Large Send Offload (LSO). This high performance NIC feature allows the host OS to pass large chunks of data payload to the NIC with instructions on how to segment the payload into multiple packets for transmission.
ULP	Upper Layer Protocol — The protocol layer above the protocol layer currently being referenced. For example, for IP the ULP is TCP.
VEB	Virtual Ethernet Bridge — This is an IEEE EVB term. A VEB is a VLAN Bridge internal to the E610 that bridges the traffic of multiple VSIs over an internal virtual network.
VEPA	Virtual Ethernet Port Aggregator — This is an IEEE EVB term. A VEPA multiplexes the traffic of one or more VSIs onto a single E610 Ethernet port. The biggest difference between a VEB and a VEPA is that a VEB can switch packets internally between VSIs, whereas a VEPA cannot.
VF	Virtual Function — A VF is a PCI Function that supports the PCI-SIG <i>Single Root I/O Virtualization and Sharing Specification</i> . One or more VFs are associated with a single PF. The group of VFs/PFs shares one or more physical resources, such as an Ethernet port. A VF is designed to be programmed by a Virtual Machine, whereas a PF is designed to be programmed by a higher-privileged entity, such as a Hypervisor.
VDMq	Virtual Machine Devices queues — VMDq is a collection of NIC features designed to offload the hypervisor from performing classification and distribution of received packets to the Virtual Machines under its control. There are two versions of VMDq: VMDq1 and VMDq2. VMDq2 is a superset of VMDq1. Its major new features are: internal switching from a Transmit Queue to a Receive Queue, the ability to replicate received multicast and broadcast packets to multiple Receive Queues, the ability to sort received packets based on a combination VLAN tag and MAC Address filter, and anti-spoofing transmit filters.

Table 15-1. Glossary and Acronyms [continued]

Acronym/Term	Description
VMM	Virtual Machine Monitor — A software component that allocates/exports/isolates server resources to the Virtual Machines (aka System Images). Other terms for VMM in this specification: Hypervisor, Virtualization Intermediary (VI).
VPD	Vital Product Data — VPD is defined in the <i>PCI Local Bus Specification</i> (this is the conventional PCI spec, not PCIe). VPD is information that uniquely identifies hardware and software elements of a server. The VPD provides a server OS with information on various Field Replaceable Units such as part number, serial number, etc.
VSI	Virtual Station Interface — This is an IEEE EVB term that defines the properties of a virtual machine's (or a physical machine's) connection to the network. Each downstream v-port on a E610 VEB or VEPA defines a VSI. A standards-based definition of VSI properties enables network management tools to perform virtual machine migration and associated network re-configuration in a vendor-neutral manner.
WoL	Wake on LAN — Wake on LAN is an Ethernet technology that enables a computer to power on or “wake up” upon receipt of a network message, often called a “magic packet”. Newer specifications in this area call <i>WoL Advanced Power Management wake up</i> .
WQ	Work Queue — One of either a Send Queue or Receive Queue.
WQE	Work Queue Element — A Work Request transformed by software into E610-specific format, and enqueued on a Work Queue.
WR	Work Request — A request to the E610 by the Consumer to perform some operation. Gets transformed by software into a Work Queue Element.
WRR	Weighted Round-Robin — A scheduling or arbitration algorithm which selects amongst requesters in a round-robin fashion, and in which each requester can be programmed to receive a different percentage share of resource bandwidth.
WSP	Weighted Strict Priority — A scheduling or arbitration algorithm which selects the requester with highest priority that has not exhausted its programmed percentage share of resource bandwidth.

Appendix A Packet Formats

A.1 Legacy Packet Formats

A.1.1 ARP Packet Formats

A.1.1.1 ARP Request Packet

Offset	# of Bytes	Field	Value	Action
0	6	Destination Address		Compare
6	6	Source Address		Stored
12	E=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	0x8100 **** 0x893F *****	Ignore
12 + E	S=(0/4)	Possible VLAN Tag		Stored
12 + E + S	2	Type	0x0806	Compare
14 + E + S	2	Hardware Type	0001	Compare
16 + E + S	2	Protocol Type	0x0800	Compare
18 + E + S	1	Hardware Size	06	Compare
19 + E + S	1	Protocol Address Length	04	Compare
20 + E + S	2	Operation	0001	Compare
22 + E + S	6	Sender Hardware Address	-	Stored
28 + E + S	4	Sender IP Address	-	Stored
32 + E + S	6	Target Hardware Address	-	Ignore
38 + E + S	4	Target IP Address	ARP IP Address	Compare

A.1.1.2 ARP Response Packet

Offset	# of Bytes	Field	Value
0	6	Destination Address	ARP Request Source Address
6	6	Source Address	Programmed from NVM or BMC
12	E=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	
12 + E	S=(0/4)	Possible VLAN Tag	From ARP Request
12 + E + S	2	Type	0x0806
14 + E + S	2	Hardware Type	0x0001
16 + E + S	2	Protocol Type	0x0800
18 + E + S	1	Hardware Size	0x06
19 + E + S	1	Protocol Address Length	0x04
20 + E + S	2	Operation	0x0002
22 + E + S	6	Sender Hardware Address	Programmed from NVM or BMC
28 + E + S	4	Sender IP Address	Programmed from NVM or BMC
32 + E + S	6	Target Hardware Address	ARP Request Sender Hardware Address
38 + E + S	4	Target IP Address	ARP Request Sender IP Address

A.1.1.3 Gratuitous ARP Packet

Offset	# of Bytes	Field	Value
0	6	Destination Address	Broadcast address.
6	6	Source Address	
12	E=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	
12 + E	S=(0/4)	Possible VLAN Tag	
12 + E + S	D=(0/8)	Possible Length + LLC/SNAP Header	
12 + E + S + D	2	Type	0x0806
14 + E + S + D	2	Hardware Type	0x0001
16 + E + S + D	2	Protocol Type	0x0800
18 + E + S + D	1	Hardware Size	0x06
19 + E + S + D	1	Protocol Address Length	0x04
20 + E + S + D	2	Operation	0x0001
22 + E + S + D	6	Sender Hardware Address	
28 + E + S + D	4	Sender IP Address	
32 + E + S + D	6	Target Hardware Address	
38 + E + S + D	4	Target IP Address	

A.1.2 IP and TCP/UDP Headers for TSO

This section outlines the format and content for the IP, TCP and UDP headers. The E610 requires baseline information from the device driver to construct the appropriate header information during the segmentation process.

Header fields that are modified by the E610 are highlighted in the figures below.

Note: The IP header is first shown in the traditional (i.e. RFC 791) representation, and because byte and bit ordering is confusing in that representation, the IP header is also shown in Little Endian format. The actual data is fetched from memory in Little Endian format.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Version				IP Hdr Length				TYPE of service								Total length (IP header + payload length)															
Identification																Flags				Fragment Offset											
Time to Live								Layer 4 Protocol ID								Header Checksum															
Source Address																															
Destination Address																															
Options																															

Figure A-1. IPv4 Header (Traditional Representation - Most Left Byte First On the Wire)

Byte 3								Byte 2								Byte 1								Byte 0															
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0								
Total length																TYPE of service								Version				IP Hdr Length											
LSB								MSB								Identification								MSB															
Fragment Offset Low								R	E	S	N	F	M	F	Fragment Offset High																								
Header Checksum																Layer 4 Protocol ID								Time to Live															
LSB																Source Address																MSB							
LSB																Destination Address																MSB							
Options																																							

Figure A-2. IPv4 Header (Little Endian Order - Byte 0 First On the Wire)

Identification is increased on each packet.

Flags Field Definitions:

The Flags field is defined below. Note that hardware does not evaluate or change these bits.

- MF - More Fragments
- NF - No Fragments
- Reserved

The E610 does TCP segmentation, not IP Fragmentation. IP Fragmentation may occur in transit through a network's infrastructure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31										
Version				Priority				Flow Label																																	
Payload Length (excluding the IP header length)																Next Header Type						Hop Limit																			
LSB																Source Address																MSB									
LSB																Destination Address																MSB									
Extensions (if any)																																									

Figure A-3. IPv6 Header (Traditional Representation - Most Left Byte First On the Wire)

Byte 3								Byte 2								Byte 1								Byte 0															
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0								
Source Address																Destination Address																							
Hop Limit								Next Header Type								Payload Length (excluding the IP header length)																							
Source Address																Destination Address																							
Source Address																Destination Address																							
Extensions																																							

Figure A-4. IPv6 Header (Little Endian Order - Byte 0 First On the Wire)

A TCP or UDP frame uses a 16-bit wide one's complement checksum. The checksum word is computed on the outgoing TCP or UDP header and payload, and on the Pseudo Header. Details on checksum computations are provided in [Section 7.2.4.6](#).

Note: TCP and UDP over IPv6 requires the use of checksum, where it is optional for UDP over IPv4.

Note: The TCP header is first shown in the traditional (i.e. RFC 793) representation, and because byte and bit ordering is confusing in that representation, the TCP header is also shown in Little Endian format. The actual data is fetched from memory in Little Endian format.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Source Port																Destination Port															
Sequence Number																															
Acknowledgment Number																															
TCP Header Length				Reserved				U R G	A C K	P S H	R S T	S Y N	F I N	Window																	
Checksum																Urgent Pointer															
Options																															

Figure A-5. TCP Header (Traditional Representation)

Byte 3								Byte 2								Byte 1								Byte 0																							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0																
Destination Port																Source Port																															
LSB																Sequence Number																MSB															
Acknowledgment Number																																															
Window																RES	URG	ACK	PSH	RST	SYN	FIN	TCP Header Length				Reserved																				
Urgent Pointer																Checksum																															
Options																																															

Figure A-6. TCP Header (Little Endian)

The TCP header is always a multiple of 32-bit words. TCP options may occupy space at the end of the TCP header and are a multiple of 8 bits in length. All options are included in the checksum.

The checksum also covers a 96-bit pseudo header prefixed to the TCP Header (see [Figure A-7](#) below). For IPv4 packets, this pseudo header contains the IP Source Address, the IP Destination Address, the IP Protocol field, and TCP Length. Software pre-calculates the partial pseudo header sum, that includes IPv4 SA, DA and protocol types, but NOT the TCP length, and stores this value into the TCP *Checksum* field of the packet. For both IPv4 and IPv6, hardware needs to factor in the TCP length to the software supplied pseudo header partial checksum.

Note: When calculating the TCP pseudo header, the byte ordering can be tricky. One common question is whether the *Protocol ID* field is added to the “lower” or “upper” byte of the 16-bit sum. The *Protocol ID* field should be added to the least significant byte (LSB) of the 16-bit pseudo header sum, where the most significant byte (MSB) of the 16-bit sum is the byte that corresponds to the first checksum byte out on the wire.

The *TCP Header Length* field is the TCP header length including option fields plus the data length in bytes, which is calculated by hardware on a frame-by-frame basis. The TCP Length does not count the 12 bytes of the pseudo header. The TCP length of the packet is determined by hardware as:

$$\text{TCP Length} = \min(\text{MSS}, \text{PAYLOADLEN}) + \text{L5\_LEN}$$

The two flags that may be modified are defined as:

- **PSH** — Receiver should pass this data to the app without delay.
- **FIN** — Sender is finished sending data.

The handling of these flags is described in [Section 7.2.4.7](#).

“Payload” is normally MSS except for the last packet where it represents the remainder of the payload.

IPv4 Source Address															
IPv4 Destination Address															
Zero				Layer 4 Protocol ID				TCP/UDP Length							

Figure A-7. TCP/UDP Pseudo Header Content for IPv4 (Traditional Representation)

IPv6 Source Address	
IPv6 Final Destination Address	
TCP/UDP Packet Length	
Zero	Next Header

Figure A-8. TCP/UDP Pseudo Header Content for IPv6 (Traditional Representation)

Notes: From RFC2460:

- If the IPv6 packet contains a Routing header, the Destination Address used in the pseudo-header is that of the final destination. At the originating node, that address is in the last element of the Routing header; at the recipient(s), that address is in the *Destination Address* field of the IPv6 header.
- The Next Header value in the pseudo-header identifies the upper-layer protocol (e.g., 6 for TCP, or 17 for UDP). It differs from the Next Header value in the IPv6 header if there are extension headers between the IPv6 header and the upper-layer header.
- The Upper-Layer Packet Length in the pseudo-header is the length of the upper-layer header and data (e.g., TCP header plus TCP data). Some upper-layer protocols carry their own length information (e.g., the *Length* field in the UDP header); for such protocols, that is the length used in the pseudo- header. Other protocols (such as TCP) do not carry their own length information, in which case the length used in the pseudo-header is the Payload Length from the IPv6 header, minus the length of any extension headers present between the IPv6 header and the upper-layer header.
- Unlike IPv4, when UDP packets are originated by an IPv6 node, the UDP checksum is not optional. That is, whenever originating a UDP packet, an IPv6 node must compute a UDP checksum over the packet and the pseudo-header, and, if that computation yields a result of zero, it must be changed to hex FFFF for placement in the UDP header. IPv6 receivers must discard UDP packets containing a zero checksum, and should log the error.

A type 0 Routing header has the following format:

Next Header	Hdr Ext Len	Routing Type "0"	Segments Left "n"
Reserved			
Address[1]			
Address[2]			
...			
Final Destination Address [n]			

Figure A-9. IPv6 Routing Header (Traditional Representation)

- **Next Header** — 8-bit selector. Identifies the type of header immediately following the Routing header. Uses the same values as the IPv4 protocol field [RFC-1700 et seq.].
- **Hdr Ext Len** — 8-bit unsigned integer. Length of the Routing header in 8-octet units, not including the first 8 octets. For the Type 0 Routing header, Hdr Ext Len is equal to two times the number of addresses in the header.
- **Routing Type** — 0.
- **Segments Left** — 8-bit unsigned integer. Number of route segments remaining (i.e., number of explicitly listed intermediate nodes still to be visited before reaching the final destination). Equal to "n" at the source node.

- **Reserved** — 32-bit reserved field. Initialized to zero for transmission; ignored on reception.
- **Address[1...n]** — Vector of 128-bit addresses, numbered 1 to n.

The UDP header is always 8 bytes in size with no options.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Source Port																Destination Port															
Length																Checksum															

Figure A-10. UDP Header (Traditional Representation)

Byte 3								Byte 2								Byte 1								Byte 0							
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
Destination Port																Source Port															
Checksum																Length															

Figure A-11. UDP Header (Little Endian Order)

UDP pseudo header has the same format as the TCP pseudo header. The pseudo header prefixed to the UDP header contains the IPv4 source address, the IPv4 destination address, the IPv4 protocol field, and the UDP length (same as the TCP Length discussed above). This checksum procedure is the same as is used in TCP.

Unlike the TCP checksum, the UDP checksum is optional (for IPv4). Software must set the *TXSM* bit in the TCP/IP Context Transmit Descriptor to indicate that a UDP checksum should be inserted. Hardware does not overwrite the UDP checksum unless the *TXSM* bit is set.

A.1.3 Magic Packet

The Magic Packet is a broadcast frame, but could also be a Multicast or Unicast Ethernet MAC Addresses. The E610 accepts this packet if it matches any of its pre-programmed Ethernet MAC Addresses. Magic packet can be sent over a variety of connectionless protocols (usually UDP or IPX). The Magic packet pattern is composed of the following sequences:

- Synchronization stream composed of 6 bytes equal to 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
- Unique pattern composed of 16 times the end node Ethernet MAC Address. The E610 expect for the Ethernet MAC Address stored in the RAL[0] and RAH[0] registers.

The E610 looks for the synchronization pattern and the sequence of 16 Ethernet MAC Addresses. It does not check the packet content and the length of the header that precedes the magic pattern nor any data that follows it.

A.2 Packet Types for Packet Split Filtering

The following packet types are supported by the 'Packet Split' feature in the E610. This section describes the packets from the 'split-header' point of view. This means that when describing the different fields that are checked and compared, it emphasizes only the fields that are needed to calculate the header length. This document describes the checks that are done after the decision to pass the packet to the host memory was done.

Terminology:

- **Compare** — The field values are compared and must be exactly equal to the value specified in this document.
- **Checked** — The field values are checked for calculation (header length ...).
- **Ignore** — The field values are ignored but the field is counted as part of the header.

A.2.1 Type 1.1: Ethernet (VLAN/SNAP) IP packets

A.2.1.1 Type 1.1: Ethernet, IP, Data

This type contains only Ethernet header and IPv4 header while the payload header of the IP is not IPv6/TCP/UDP.

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – Processed by main address filter, or broadcast.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN, E-Tag)	0x8100 **** 0x893F *****	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100 ****	Compare	
12+D+S	D=(0/8)	Possible Length + LLC/ SNAP Header		Compare	
12+D+S	2	Type	0800h	Compare	IP
IPv4 Header					
14+D+S	1	Version/ HDR length	0x4X	Compare	Check IPv4 and header length.
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	>0 or MF bit is set	Check	Check that the packet is fragmented.
22+D+S	1	Time to live	-	Ignore	
23+D+S	1	Protocol		Ignore	Has no meaning if the packet is fragmented.
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	

Offset	# of Bytes	Field	Value	Action	Comment
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	

A.2.1.2 Type 1.2: Ethernet (SNAP/VLAN), IPv4, UDP

This type contains only Ethernet header, IPv4 header, and UDP header.

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – Processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	0x8100 **** 0x893F *****	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100 ****	Check	
12+D+S	2	Type	0x0800	Compare	IP
IP Header					
14+D+S	1	Version/ HDR length	0x4X	Compare	Check IPv4 and header length.
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	(xx00) 0x000	Compare	
22+D+S	1	Time to live	-	Ignore	
23+D+S	1	Protocol	0x11	Compare	UDP header.
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	
UDP Header					
34+D+S+N	2	Source Port	Not (0x801)	Check	Not NFS packet.
36+D+S+N	2	Destination Port	Not (0x801)	Check	Not NFS packet.
38+D+S+N	2	Length	-	Ignore	
40+D+S+N	2	Checksum	-	Ignore	

In this case, the packet is split after (42+D+S+N) bytes.

A.2.1.3 Type 1.3: Ethernet (VLAN/SNAP) IPv4 TCP

This type contains only Ethernet header, IPv4 header, and TCP header.

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – Processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	0x8100 **** 0x893F *****	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100 ****	Check	
12+D+S	2	Type	0x0800	Compare	IP
IPv4 Header					
14+D+S	1	Version/ HDR length	0x4X	Compare	Check IPv4 and header length.
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	0x00	Compare	
22+D+S	1	Time to live	-	Ignore	
23+D+S	1	Protocol	0x06	Compare	TCP header.
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	
TCP Header					
34+D+S+N	2	Source Port	Not (0x801)	Check	Not NFS packet.
36+D+S+N	2	Destination Port	Not (0x801)	Check	Not NFS packet.
38+D+S+N	4	Sequence number	-	Ignore	
42+D+S+N	4	Acknowledge number	-	Ignore	
46+D+S+N	1/2	Header Length		Check	
46.5+D+S+N	1.5	Different bits	-	Ignore	
48+D+S+N	2	Window size	-	Ignore	
50+D+S+N	2	TCP checksum	-	Ignore	
52+D+S+N	2	Urgent pointer	-	Ignore	
54+D+S+N	F	TCP options	-	Ignore	

In this case, the packet is split after (54+D+S+N+F) bytes.

- $N = (\text{IP HDR length} - 5) * 4$.
- $F = (\text{TCP header length} - 5) * 4$.

A.2.1.4 Type 1.4: Ethernet IPv4 IPv6

A.2.1.4.1 IPv6 Header Options Processing

This type of processing looks at the next-header field and header length to determine the identity of the next-header processes, the IPv6 options, and its length.

If the next header in the IPv6 header is equal to 0x00/0x2B/0x2C/0x3B/0x3C it means that the next header is an IPv6 option header and this is its structure:

Next Header (8 bit)	Header Len (8 bit)	
Option Header Parameters		

Header Len determines the length of the header while the next header field determines the identity of the next header (should be any IPv6 extension header for another IPv6 header option).

A.2.1.4.2 IPv6 Next Header Values

When parsing an IPv6 header, the E610 does not parse all possible extension headers. If there is an extension header that is not supported by the E610, the packet is treated as an unknown payload after the IPv6 header.

Value	Header Type
0x00	Hop by Hop
0x2B	Routing
0x2C	Fragment
0x3B	No next header (EOL)
0x3C	Destination option header

The next header in a fragment header is ignored and this extension header is expected to be the last header.

A.2.1.4.3 Type 1.4.1: Ethernet IPv4 IPv6 Data

This type contains only Ethernet header, IPv4 header, and IPv6 header.

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – Processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	0x8100 **** 0x893F *****	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100	Check	
12+D+S	2	Type	0x0800	Compare	IP

Offset	# of Bytes	Field	Value	Action	Comment
IPv4 Header					
14+D+S	1	Version/ HDR length	0x4X	Compare	Check IPv4 and header length.
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	0x00	Compare	
22+D+S	1	Time to live	-	Ignore	
23+D+S	1	Protocol	0x29	Compare	IPv6
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	
IPv6 Header					
34+D+S+N	1	Version/ Traffic Class	0x6X	Compare	Check IPv6.
35+D+S+N	3	Traffic Class/Flow Label	-	Ignore	
38+D+S+N	2	Payload Length	-	Ignore	
40+D+S+N	1	Next Header	IPv6 extension headers	Check	
41+D+S+N	1	Hop Limit	-	Ignore	
42+D+S+N	16	Source Address	-	Ignore	
48+D+S+N	16	Destination Address		Ignore	
74+D+S+N	B	Possible IPv6 Next Headers	-	Ignore	

In this case the packet is split after (74+D+S+N+B) bytes.

- $N = (\text{IP HDR length} - 5) * 4$.
- One of the extension headers of the IPv6 packets must be a "fragment header" for the packet to be parsed.

A.2.1.4.4 Type 1.4.2: Ethernet (VLAN/SNAP) IPv4 IPv6 UDP

This type contains only Ethernet header, IPv4 header, IPv6 header and UDP header.

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – Processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	0x8100 **** 0x893F *****	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100	Check	
12+D+S	2	Type	0x0800	Compare	IP

Offset	# of Bytes	Field	Value	Action	Comment
IPv4 Header					
14+D+S	1	Version/ HDR length	0x4X	Compare	Check IPv4 and header length.
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	0x00	Compare	
22+D+S	1	Time to live	-	Ignore	
23+D+S	1	Protocol	0x29	Compare	IPv6
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	
IPv6 Header					
34+D+S+N	1	Version/ Traffic Class	0x6X	Compare	Check IPv6.
35+D+S+N	3	Traffic Class/Flow Label	-	Ignore	
38+D+S+N	2	Payload Length	-	Ignore	
40+D+S+N	1	Next Header	IPv6 extension header or 0x11	Check	IPv6 extension headers.
41+D+S+N	1	Hop Limit	-	Ignore	
42+D+S+N	16	Source Address	-	Ignore	
58+D+S+N	16	Destination Address		Ignore	
74+D+S+N	B	Possible IPv6 Next Headers	-	Ignore	
UDP Header					
74+D+S+N+B	2	Source Port	Not (0x801)	Check	Not NFS packet.
76+D+S+N+B	2	Destination Port	Not (0x801)	Check	Not NFS packet.
78+D+S+N+B	2	Length	-	Ignore	
80+D+S+N+B	2	Checksum	-	Ignore	

In this case the packet is split after (82+D+S+N+B) bytes.

$N = (\text{IP HDR length} - 5) * 4.$

A.2.1.4.5 Type 1.4.3: Ethernet (VLAN/SNAP) IPv4 IPv6 TCP

This type contain only Ethernet header, IPv4 header, IPv6 header and TCP header.

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – Processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	0x8100 **** 0x893F *****	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100	Check	
12+D+S	2	Type	0x0800	Compare	IP
IPv4 Header					
14+D+S	1	Version/ HDR length	0x4X	Compare	Check IPv4 and header length
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	0x00	Compare	
22+D+S	1	Time to live	-	Ignore	
23+D+S	1	Protocol	0x2A	Compare	IPv6
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	
IPv6 Header					
34+D+S+N	1	Version/ Traffic Class	0x6X	Compare	Check IPv6
35+D+S+N	3	Traffic Class/Flow Label	-	Ignore	
38+D+S+N	2	Payload Length	-	Ignore	
40+D+S+N	1	Next Header	IPv6 extension header Or 0x06	Check	IPv6 extension headers
41+D+S+N	1	Hop Limit	-	Ignore	
42+D+S+N	16	Source Address	-	Ignore	
58+D+S+N	16	Destination Address		Ignore	
74+D+S+N	B	Possible IPv6 Next Headers	-	Ignore	
TCP Header					
74+T	2	Source Port	Not (0x801)	Check	Not NFS packet
76+T	2	Destination Port	Not (0x801)	Check	Not NFS packet
78+T	4	Sequence number	-	Ignore	
82+T	4	Acknowledge number	-	Ignore	
86+T	1/2	Header Length		Check	
86.5+T	1.5	Different bits	-	Ignore	

Offset	# of Bytes	Field	Value	Action	Comment
88+T	2	Window size	-	Ignore	
90+T	2	TCP checksum	-	Ignore	
92+T	2	Urgent pointer	-	Ignore	
94+T	F	TCP options	-	Ignore	

In this case the packet is split after (94+D+S+N+B+F) bytes.

- $T = D + S + N + B$
- $N = (\text{IP HDR length} - 5) * 4$.
- $F = (\text{TCP HDR length} - 5) * 4$

A.2.2 Type 2: Ethernet, IPv6

A.2.2.1 Type 2.1: Ethernet, IPv6 Data

This type contains only an Ethernet header and an IPv6 header while the packet should be a fragmented packet. If the packet is not fragmented and the Next header is not supported, the header is not split. The supported packet types for header split are programmed in PSRTYPE register (per VF).

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – Processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	0x8100 **** 0x893F *****	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100	Check	
IPv6 Header					
12+D+S	2	Type	0x86DD	Compare	IP
14+D+S	1	Version/ Traffic Class	0x6X	Compare	Check IPv6
15+D+S	3	Traffic Class/Flow Label	-	Ignore	
18+D+S	2	Payload Length	-	Ignore	
20+D+S	1	Next Header	IPv6 next header types	Check	The last header must be fragmented header for the header to be split.
21+D+S	1	Hop Limit	-	Ignore	
22+D+S	16	Source Address	-	Ignore	
38+D+S	16	Destination Address		Ignore	
54+D+S	N	Possible IPv6 Next Headers	-	Ignore	

In this case the packet is split after (54+D+S+N) bytes.

- The last next header field of the IP section field should not be 0x11/0x06 (TCP/UDP).

A.2.2.1.1 Type 2.2: Ethernet (VLAN/SNAP) IPv6 UDP

This type contains only Ethernet header, IPv6 header, and UDP header.

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – Processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	0x8100 **** 0x893F *****	Ignore	
12+D	S=(0/4)	Possible VLAN Tag		Check	
IPv6 Header					
12+D+S	2	Type	0x86DD	Compare	IP
14+D+S	1	Version/ Traffic Class	0x6X	Compare	Check IPv6.
15+D+S	3	Traffic Class/Flow Label	-	Ignore	
18+D+S	2	Payload Length	-	Ignore	
20+D+S	1	Next Header	IPv6 next header types Or 0x11	Check	
21+D+S	1	Hop Limit	-	Ignore	
22+D+S	16	Source Address	-	Ignore	
38+D+S	16	Destination Address		Ignore	
54+D+S	N	Possible IPv6 Next Headers	-	Ignore	
UDP Header					
54+D+S+N	2	Source Port	Not (0x801)	Check	Not NFS packet.
56+D+S+N	2	Destination Port	Not (0x801)	Check	Not NFS packet.
58+D+S+N	2	Length	-	Ignore	
60+D+S+N	2	Checksum	-	Ignore	

In this case the packet is split after (62+D+S+N) bytes.

- The last 'next-header' field of the last header of the IP section must be 0x06.

A.2.2.2 Type 2.3: Ethernet (VLAN/SNAP) IPv6 TCP

This type contains only Ethernet header, IPv6 header, and UDP header.

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – Processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	0x8100 **** 0x893F *****	Ignore	
12+D	S=(0/4)	Possible VLAN Tag		Check	
IPv6 Header					
12+D+S	2	Type	0x86DD	Compare	IP
14+D+S	1	Version/ Traffic Class	0x6X	Compare	Check IPv6
15+D+S	3	Traffic Class/Flow Label	-	Ignore	
18+D+S	2	Payload Length	-	Ignore	
20+D+S	1	Next Header	IPv6 next header types Or TCP	Check	
21+D+S	1	Hop Limit	-	Ignore	
22+D+S	16	Source Address	-	Ignore	
38+D+S	16	Destination Address		Ignore	
54+D+S	N	Possible IPv6 Next Headers	-	Ignore	
TCP Header					
54+D+S+N	2	Source Port	Not (0x801)	Check	Not NFS packet
56+D+S+N	2	Destination Port	Not (0x801)	Check	Not NFS packet
58+D+S+N	4	Sequence number	-	Ignore	
62+D+S+N	4	Acknowledge number	-	Ignore	
66+D+S+N	1/2	Header Length		Check	
66.5+D+S+N	1.5	Different bits	-	Ignore	
68+D+S+N	2	Window size	-	Ignore	
70+D+S+N	2	TCP checksum	-	Ignore	
72+D+S+N	2	Urgent pointer	-	Ignore	
74+D+S+N	F	TCP options	-	Ignore	

In this case the packet is split after (54+D+S+N+F) bytes.

- $F = (\text{TCP header length} - 5) * 4$.
- The last 'next-header' field of the last header of the IP section must be 0x11.

A.2.3 Type 3: Reserved

Type 3 used to be iSCSI packets (Header split is not supported for iSCSI packets in the E610).

A.2.4 Type 4: Reserved

A.2.5 Type 5: Cloud Packets

A.2.5.1 Type 5.1: Ethernet, IPv4, NVGRE, IPv4/6, TCP/UDP

This type contains an Ethernet header an IPv4 Header, a GRE header an IPv4/6 header and a TCP or UDP header. The supported packet types for header split are programmed in PSRTYPE register (per VF).

Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – Processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	0x8100 **** 0x893F *****	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100	Check	
12+D+S	2	Type	0x0800	Compare	IP Split on outer L2 is done after this field.
IPv4 Header					
14+D+S	1	Version/ HDR length	0x4X	Compare	Check IPv4 and header length.
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	0x00	Compare	
22+D+S	1	Time to live	-	Ignore	
23+D+S	1	Protocol	0x2F	Compare	GRE
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	
GRE Header					
34 + D + S + N	2	Flags/Version	0x2000	Compare	
36 + D + S + N	2	Protocol Type	0x6558	Compare	
38+ D + S + N	3	TNI		Store	Used for Flow director.
41 + D + S + N	1	Reserved	0x0	Ignore	Split on Cloud header splits after this field.

Offset	# of Bytes	Field	Value	Action	Comment
Inner MAC Header					
42 + D + S + N	6	Inner Destination Address		Ignore	Used for Flow director.
48 + D + S + N	6	Inner Source Address		Ignore	
54 + D + S + N	I=(0/4)	Possible Inner VLAN Tag	0x8100	Store	Used for Flow director.
54 + D + S + N + I	2	Type	0x0800/0x86DD	Compare	Split on L2 is done after this field.
IPv4/6 header - as described above					Split on L3 is done after these fields.
UDP/TCP - as described above					Split on L4 is done after these fields.

A.2.5.2 Type 5.2: Ethernet, IPv4, VXLAN, MAC, IPv4/6, TCP/UDP

This type contains an Ethernet header an IPv4 Header, a UDP header with a specific UDP destination port a VXLAN header, an inner MAC header, an IPv4/6 header and a TCP or UDP header. The supported packet types for header split are programmed in PSRTYPE register (per VF).

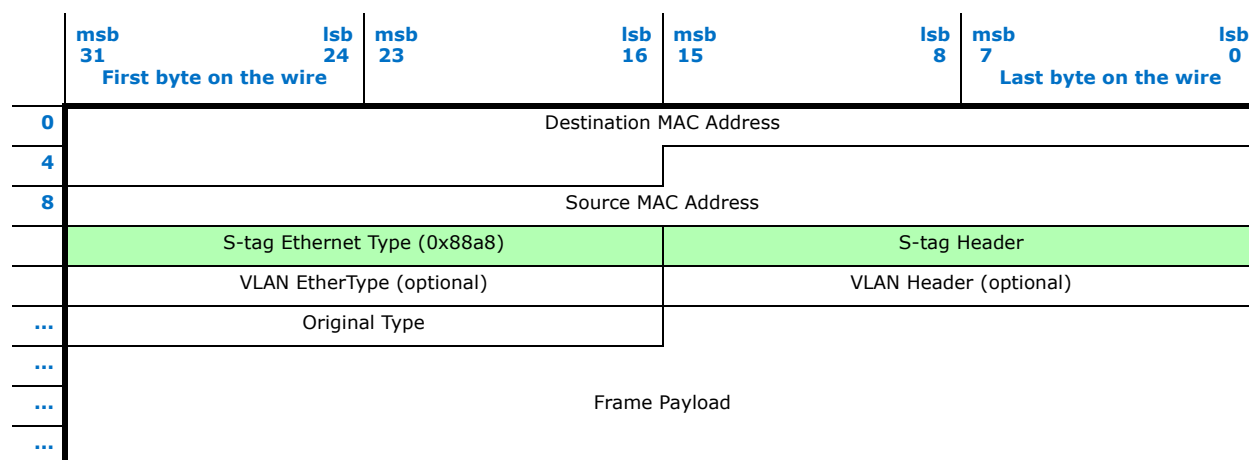
Offset	# of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – Processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN, E-tag)	0x8100 **** 0x893F *****	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100	Check	
12+D+S	2	Type	0x0800	Compare	IP Split on outer L2 is done after this field
IPv4 Header					
14+D+S	1	Version/ HDR length	0x4X	Compare	Check IPv4 and header length
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	0x00	Compare	
22+D+S	1	Time to live	-	Ignore	
23+D+S	1	Protocol	0x11	Compare	UDP
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	

Offset	# of Bytes	Field	Value	Action	Comment
UDP Header					
34+D+S+N	2	Source Port		Ignore	
36+D+S+N	2	Destination Port		Compare	Compare to VXLANCN-TRL.UDPPORT
38+D+S+N	2	Length	-	Ignore	
40+D+S+N	2	Checksum	-	Ignore	
VXLAN Header					
42 + D + S + N	1	Flags	0x08	Compare	VNI exists
43 + D + S + N	3	Reserved	0x000000	Compare	
46 + D + S + N	3	VNI		Store	Used for flow director
49 + D + S + N	1	Reserved	0x00	Compare	Split on Cloud header splits after this field.
Inner MAC Header					
50 + D + S + N	6	Inner Destination Address		Ignore	Used for Flow director
56 + D + S + N	6	Inner Source Address		Ignore	
62 + D + S + N	I=(0/4)	Possible Inner VLAN Tag	0x8100	Store	Used for Flow director
62 + D + S + N + I	2	Type	0x0800/0x86DD	Compare	Split on L2 is done after this field
IPv4/6 header - as described above					Split on L3 is done after these fields
UDP/TCP - as described above					Split on L4 is done after these fields

A.3 E-tag and S-tag Formats

Note: S-tags are not supported in the E610.

Packets with S-tag has the following format:



The S-tag format is the format defined for S-tags in the IEEE 802.1ad specification as described below:

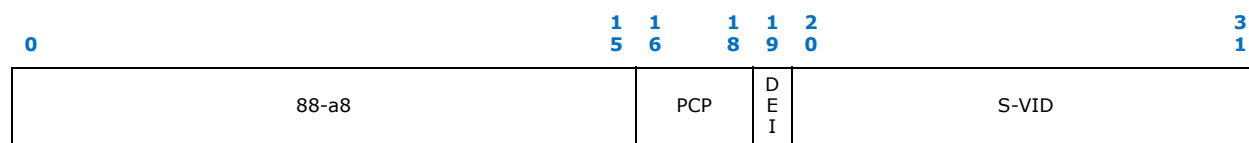
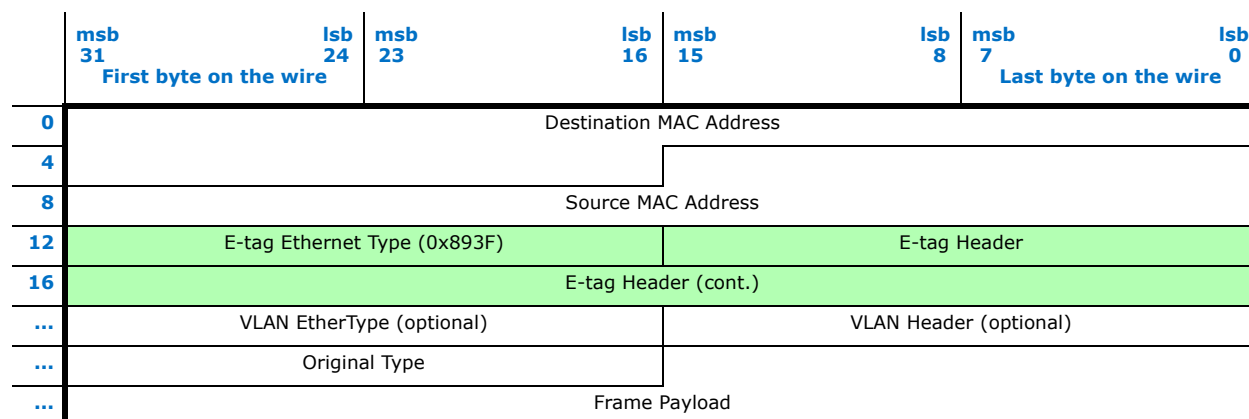


Figure A-12. S-tag Format

Packets with E-tag has the following format:





The E-tag format is the format defined in the IEEE 802.1BR specification as described below:

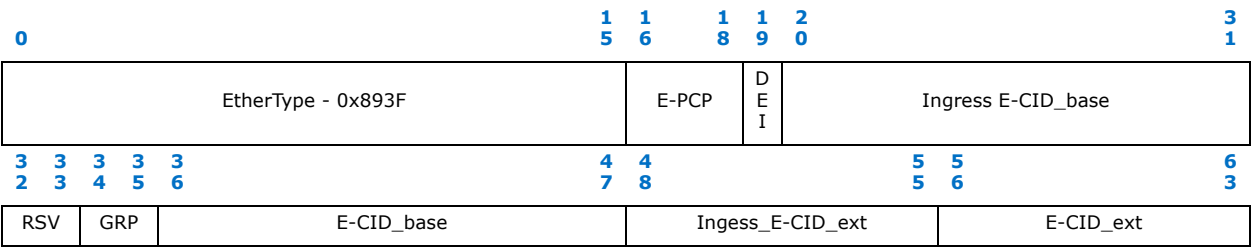


Figure A-13. E-tag Format

The Ingress_E-CID_ext and E-CID_ext are always zero for endpoints and are effectively reserved.

Appendix B Manufacturing

B.1 Introduction

OEM Provisioning is the broad term used to describe the overall process through which customers (i.e., OEMs, ODMs, end customers, and any third-party contractors involved in the device production life cycle) configure the device for initial production deployment.

The following sections describe the firmware requirements enabling these provisioning steps, the firmware back-end interface to the customer-facing tools, and the associated security requirements. In E610, the provisioning by the OEM is limited to only the SPDM Device ID Public Key certificates.

Hardware/EMP Firmware blocks involved during OEM provisioning flows:

- NVM — R/W access to the NOR SPI flash.
- EMP — Contains the HAL for blocks such as the SBB/NVM and higher firmware policy layers, as well as the Admin command interface used by the PFs and Tools for device configuration.

The overall end users of the manufacturing/OEM provisioning admin commands and the associated host tools (for example, NVUpdate for in-field, and EEUpdate for board MFG) are:

- Intel/ODM board manufacturing for provisioning SPDM certificates and completing the fusing of devices for Intel-owned image components.
 - In this case, Intel builds both the devices themselves AND installs them on boards.
 - The end Intel product in this case is a complete Network Interface Card (NIC).
- OEMs or end customers who design custom boards using Intel-provided ASICs.
 - In this case, Intel builds the devices, while the OEM installs them on boards (in either NIC or LOM configurations).
 - The end Intel product in this case is the chip itself.

OEMs or end customers who purchase devices (as opposed to NICs) from Intel, are required to follow the steps to complete the provisioning of SPDM certificates to the devices. Not doing so can result in the firmware not reporting CERT_CAP as part of SPDM capabilities. While SPDM will function, the REQUESTER must have out-of-band knowledge of the E610 device public key to perform SPDM attestation.

B.2 Intel Devices

Intel-branded NICs, built by Intel and sold as complete boards, always meet the following conditions and limitations:

- The Class Test Chip Manufacturing Flow has been carried out by Intel, with the following results:
 - Intel KAKs are provisioned to the Signature RoT (*fuse_signature_key0/1/2* are programmed with Intel key hashes).
 - Secure boot for the firmware is enabled. Only Intel-signed firmware is booted, and only Intel-signed firmware is accepted via firmware update (*fuse_authentication_enable* is set).

- The SPDM UDS is provisioned, so the parts have SPDM enabled.
- The Storage Root Keys (SRKs) are provisioned.
- FIPS self-tests are enabled (*fuse_fips_self_test_en* is set).
- All SREV (NVM, OROM, Netlist) fuse values are set to 1.
- The DfX Aggregator is fused to the Production DLCS state.
- The Configuration Signing fuses are programmed by Intel, and the Netlist and OROM are signed by Intel.
- The OEM RoT is not provisioned, and all keys are locked-out as all zeros (*fuse_oem_kak0/1/2* are all locked out as zero).
- Vendor Authorized Debug is NOT enabled (the JCD fuse is zero).
- Blank Flash Programming Mode is enabled (*fuse_bfp_dis* is NOT set).
- SREV is stored in NVM, not fuses (*fuse_srev_in_fuses* is NOT set).
- OEMs and end customers are unable to change any of the device's fuse settings. The only possible fuse programming events are the result of ISK revocation flows.
- The Board Manufacturing Flow is carried out by Intel, with the following results:
 - An SPDM certificate is provisioned to the NVM in slot 0. All other slots are unpopulated. This certificate is rooted in an Intel CA.
 - OEMs and end customers may program SPDM certificate slots 1 through 3 using the Program Certificate Slot admin queue command. Slot 0 cannot be overwritten.

B.3 OEM Devices

Devices sold in chip form, where the placement of the chip on a board is done by an OEM or end customer, meet the following conditions:

- The Class Test Chip Manufacturing Flow has been carried out by Intel, with the following results:
 - Intel KAKs are provisioned to the Signature RoT (*fuse_signature_key0/1/2* are programmed with Intel key hashes).
 - Secure boot for the firmware is enabled. Only Intel-signed firmware is booted, and only Intel-signed firmware is accepted via firmware update (*fuse_authentication_enable* is set).
 - The SPDM UDS is provisioned, so the parts have SPDM enabled.
 - The Storage Root Keys (SRKs) are provisioned.
 - FIPS self-tests are enabled (*fuse_fips_self_test_en* is set).
 - All SREV (NVM, OROM, Netlist) fuse values are set to 1.
 - The DfX Aggregator is fused to the Production DLCS state.
 - The Configuration Signing fuses are programmed by Intel, and the Netlist and OROM are signed by Intel.
 - The OEM RoT is not provisioned, and all keys are locked-out as all zeros (*fuse_oem_kak0/1/2* are all locked out as zero).
 - Vendor Authorized Debug is NOT enabled (the JCD fuse is zero).

- Blank Flash Programming Mode is enabled (*fuse_bfp_dis* is NOT set).
- SREV is stored in NVM, not fuses (*fuse_srev_in_fuses* is NOT set).
- The Board Manufacturing Flow MUST be carried out by the OEM or the end customer if SPDm is to be used with the OEM devices. If not carried out, SPDm does not support the usage of certificates, CERT_CAP is not reported, and while SPDm is functional, the REQUESTER is required to “know” the public key of the device via out-of-band provisioning.
 - At a high level: The OEM or End Customer is required to use the commands documented in this appendix to provision the Intel-provided certificate into the NVM.
 - If SPDm usage is needed, then (at a minimum) a Device ID public key certificate must be provisioned into the device's NVM (flash) by using the Program Certificate Slot command on certificate slot 0.
 - Failure to do so means that the SPDm requester has no way to obtain a certificate from the device itself using the SPDm GET_CERTIFICATE command. SPDm will still be functional, but the public key of the device must be provisioned into the requester out-of-band for SPDm to work. Typically, working without certificates is only a realistic option for LOM devices, where a replacement of the device is not feasible without de-soldering the chip.
 - An Intel-endorsed certificate is available from an Intel web endpoint.
 - Intel recommends that slot 0 is populated with the Intel certificate, with slots 1 through 3 available for the OEM or end customers. However, the OEM or end customer may also utilize their own certificate in slot 0 if desired.
 - At least slot 0 must be provisioned.
 - The Generate Distribution Point URL command returns to the caller the publicly-visible URL from which a copy of the Device ID Leaf Certificate for the device can be obtained at any time. The per-family and the overall Intel Root CA certificate URLs are not obtainable from the device itself. These are required to be provided by Intel as part of the board-level manufacturing tools, and are typically embedded in the tool that does SPDm provisioning. These certificates are required to create the full certificate chain.

B.4 Fuse Programming

There is no fuse programming available to the OEM or end customer on E610 devices.

B.5 Technical Definitions

B.5.1 Overview

At a high level, there are two overall manufacturing stages during the product life cycle before initial deployment:

- **Chip Manufacturing** — Always done by Intel.
- **Board Manufacturing** — Is done by Intel (for Intel-built NICs), or by an OEM (for custom OEM designs or for LAN-On-Motherboard usages).

B.5.1.1 Chip Manufacturing

The facility where the device undergoes class test, binning, and HVM fuse programming (for example, PVT/UID/Intel debug access provisioning, and so on). All E610 devices pass this stage, regardless of whether they are going onto Intel assembled NICs or OEM assembled boards (as NICs or LOMs).

This can be an Intel internal facility or an Intel contracted third-party.

B.5.1.2 Board Manufacturing

- **Intel Board Manufacturing** — For NICs assembled and branded by Intel, this step takes place.
 - This can be an Intel internal facility or an Intel contracted third-party (for example, ODM).
 - To complete functional testing before the devices can be shipped, the assembled board is provisioned with initial reference NVM content.
 - When the device is a mass-market SKU, Intel is the OEM and both the ODM & the OEM flows are done at Intel or it's contracted facilities.
- **OEM Board Manufacturing** — When the board is assembled externally to Intel (such as LOMs of custom OEM board designs), it is assembled at a downstream facility (this can be directly at a customer within some kind of provisioning/manufacturing step, at their third-party contractors, or at an OEM/ODM, and so on) where the device is provisioned before the production deployment.
 - This is generically referred to as "OEM Board Manufacturing" to highlight that this stage requires customers to directly control the configuration/provisioning of their devices.

Within the Manufacturing stages described above, there are three firmware states:

- **Chip Manufacturing Firmware** — Dedicated build used at Class Test for Attestation Provisioning. Not within scope for board MFG / OEM Provisioning. Cannot be executed by parts after Chip Manufacturing.
- **OEM Provisioning / Board Manufacturing** — The commands and features added to the production firmware image, but only available under certain conditions (see the command details below).
 - These are added to the production image so that a single image can be used for configuration provisioning and for later deployment, as this saves on manufacturing line time as reprogramming the flash part twice is not required.
- **Main Firmware** — The production firmware in production mode, with the board manufacturing commands disabled or limited.

B.5.1.3 Architectural Assumptions/Background

Intel recommends that the OEM Board Manufacturing environment must be trusted:

- Since the environment is trusted, it is assumed that the physical interfaces used to issue commands are secure and no additional requirements are placed for using secure channels. No unlocks are required to send the commands.

B.6 Admin Queue Command Definitions

All commands are available to PFs only, and only in normal operational mode (that is, they are not available in Firmware Recovery Mode).

These commands are available on both the PF Admin Queue and on the Tools Queue.

Table B-1. Generate Distribution Point URL (0xDD02)

Name	Opcode	Available After End of Board Manufacturing Condition Reached?	Comments
Generate Distribution Point URL	0xDD02	Yes	Can always be used to retrieve the certificate URL of the device, in case a certificate needs to be retrieved in the field (for example, due to flash corruption).
Program Certificate Slot	0xDD03	Yes	

This command is used during board manufacturing by the manufacturing environment to request the device firmware to return the URL from where specific provisioning content can be downloaded. The URL returned by the device provides the Device ID Public Key Certificate. The Device Family and IPCS DICE Root Certificates in the chain are obtained from Intel, and not from the device. It is expected that the manufacturing software tool bundles them.

Customers/OEMs may reuse this command in the field if they wish to re-download the Intel-issued Device Identity certificate for a device (for example, to authenticate and then certify the PubKey to their own authority chain for an alternate SPDH certificate slot).

Firmware sends a synchronous completion after updating the provided response buffer with the URL, or an error status indicator.

Table B-2. Command/Response Descriptor

Argument	Byte(s)	Value	Description
Flags	0-1		See Section 10.11.2.4.2.1 for details.
Opcode	2-3	0xDD02	Command opcode.
Datalen	4-5	0x0	Length of provided response buffer in bytes. This is the maximum possible size, firmware will report the actual URL length as part of the response <i>Datalen</i> field. If the buffer size is too small for the URL string, the firmware returns EFBIG. It is recommended the SW always send a 4KB buffer here as the string is guaranteed to fit within an AQ command buffer.
Return Value	6-7	0x0	Must be zeroed by the software device driver.
Cookie High	8-11	Cookie	Opaque value copied into the response.
Cookie Low	12-15	Cookie	Opaque value copied into the response.
Reserved (Command Flags)	16	0x0	Reserved. Must be 0x0. Placeholder for future command option expansion. Ignored by firmware.
Reserved	17-23	0x0	Reserved. Ignored by firmware.
Data Address High	24-27	0x0	Address of command/response buffer.
Data Address Low	28-31	0x0	Address of command/response buffer.

The output buffer contains the device's unique URL generated by the EMP firmware. Encoding/Format of the generated ASCII URL is as follows:

```
https://tsci.intel.com/content/NIC/certs/<FamilyID:8b>_<uid:64b>_<base64url(SKI.Trim(96))>:96b>.cer
```

where:

- *<FamilyID>* is the 8-bit identifier per product family written to *fuse_family_id* in the product fuses.
 - 0x02 for LKV product family.
- *<uid>* is the device's 64-bit unique identifier value (UID) in fuses (generated by Intel ISEED and programmed at class test/chip manufacturing).

Note: This is NOT the same as the per-device UDS or the DeviceID key-pair for SPDm/Attestation.

- *SKI* (Subject Key Identifier) encoded in this URL: 16 base64url text characters derived from the most significant 96-bits of the SKI.
 - SKI value itself is calculated as the most significant 160-bits of the SHA2-384 digest of the device-unique DevID public key ASN.1 BIT string (i.e., hash input includes the first flag byte indicating compression, see RFC7093 Section 2 Option 2).

Note: base64url encodes groups of 6-bits into ASCII characters.

- All of the fields are fixed width matching the specified lengths in the schema above.
- All fields are encoded as big-endian (MSB-first) into the ASCII representation in the URL.Program Certificate Slot (0xDD03)

This command is initially used during OEM board manufacturing by the manufacturing environment to program SPDm Certificate Slot 0 in the NVM with Intel's specific Device Identity certificate chain matching the device. Once the "End of Board Manufacturing" (EOBM) condition is reached, the firmware enforces a check that only slots without a valid certificate chain can be written. Customers/OEMs may use this command to program remaining slots (1-3) that are not already populated. The device does not support certificate slot indexes 4-7 and attempts to write to these are rejected.

The command must be accompanied by a buffer containing the full ASN.1 DER-encoded X.509 certificate chain from a Root CA to the leaf Device Identity certificate. The firmware validates that the specified certificate contains the a matching Subject Key as the device's unique DevID public key reported by the SBB to firmware during the SPDm initialization flow.

The firmware is responsible for appending the Certificate Chain Format header (52 bytes) from the SPDm specification to the user provided buffer before writing to the NVM slot. See "Certificate Chain Format" from the SPDm v1.1.2 specification for additional details.

The firmware sends a synchronous completion status.

The response does not provide a buffer. A return value of 0x0 (success) indicates the following, depending on the Command Flag Write/Validate indication provided in the descriptor:

- If Write was requested, the provided certificate was written to the requested certificate slot.
- If Read was requested, the requested certificate slot has a valid certificate chain matching the DeviceID public key in the leaf certificate.

If an invalid certificate slot is requested, the command returns EINVAL.

If Read was requested and the requested slot does not contain a valid certificate chain, the command returns ENXIO.

NOTE: *This page intentionally left blank.*



LEGAL

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

This document (and any related software) is Intel copyrighted material, and your use is governed by the express license under which it is provided to you. Unless the license provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this document (and related materials) without Intel's prior written permission. This document (and related materials) is provided as is, with no express or implied warranties, other than those that are expressly stated in the license.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors which may cause deviations from published specifications.

Copies of documents that are referenced in this document can be obtained by visiting www.intel.com/design/literature.htm.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

* Other names and brands may be claimed as the property of others.

© 2022-2026 Intel Corporation.